

# **Relevance Feedback in Flexible Retrieval Environment**

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

Poorva Potnis

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

September 2005

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

UNIVERSITY OF MINNESOTA

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

Poorva Potnis

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

\_\_\_\_\_  
Carolyn J. Crouch

Name of Faculty Advisor

\_\_\_\_\_  
Signature of Faculty Advisor

\_\_\_\_\_  
Date

GRADUATE SCHOOL

© Poorva Potnis 2005

## **Abstract**

Flexible retrieval systems retrieve the most relevant part(s) of a document with respect to a query, unlike traditional text search systems that retrieve entire documents. Relevance feedback is a technique to improve the results of retrieval. It uses information from the result of an initial retrieval to modify the query in such a way that more relevant documents are fetched the second time. A user may specify which documents are relevant and which are not. In pseudo feedback a query is modified using information from documents that the retrieval system assumes to be relevant.

In this thesis we study the effect of relevance feedback in a flexible retrieval environment. We experiment with both pseudo feedback and relevance feedback, using a wide range of values for the feedback parameters. Overlap is an important aspect of the flexible retrieval environment. The structured retrieval result usually contains both elements and their descendants. We consider how overlap in the elements used for feedback affects the results of retrieval.

## **Acknowledgements**

Many people have contributed towards the successful completion of this thesis.

I would like to thank Dr. Carolyn Crouch for her continuous support and guidance. I am grateful to her for giving me an opportunity to work in the exciting field of Information Retrieval and solve an interesting problem as the subject of my thesis.

I would also like to thank Dr. Donald Crouch, for his suggestions and feedback during the important stages of my thesis work.

I appreciate the invaluable help that I received from my fellow students. Many thanks to Sudip Khanna for his useful suggestions and encouragement at the right times, and to Nagendra Doddapaneni and Aniruddha Mahajan for their prompt help.

I would like to thank the faculty and staff of the Department of Computer Science at UMD, especially Steven Holtz, Jim Luttinen, Lori Lucia and Linda Meek.

Last but not the least, thanks to my family and friends at UMD for the support and encouragement they provided during the two years of this degree. I could not have accomplished it without you.

# Table of Contents

List of Tables .....	vi
List of Figures.....	vii
<b>Chapter 1. Introduction.....</b>	<b>1</b>
<b>Chapter 2. Basic Concepts of Information Retrieval .....</b>	<b>5</b>
2.1 Indexing, Weighting and Retrieval.....	5
2.2 Vector Space Model.....	7
2.3 Extended Vector Space Model.....	7
2.4 Smart Retrieval Engine.....	8
2.5 Relevance Feedback.....	8
<b>Chapter 3. Experimental Data and Evaluation Measures .....</b>	<b>10</b>
3.1 Document Collection .....	10
3.2 Query Collection.....	10
3.3 Relevance Judgments.....	12
3.4 Evaluation Measures.....	14
<b>Chapter 4. Flexible Retrieval System.....</b>	<b>16</b>
4.1 What is Flexible Retrieval .....	16
4.2 Overview of Flexible Retrieval System.....	16
4.3 Preprocessing and Indexing Documents.....	17
4.4 Retrieving and Scoring Elements.....	18
4.4.1 Retrieval of Leaf Elements .....	19
4.4.2 Populating the Documents .....	22

4.5 Design of the Flexible Retrieval System .....	23
4.5.1 Input Processing.....	24
4.5.2 Flex – the Flexible Retrieval Core.....	25
4.5.3 Smart Interface.....	27
4.5.4 Output Processing.....	27
4.6 Implementation of the Flexible Retrieval System .....	27
<b>Chapter 5. Relevance Feedback.....</b>	<b>29</b>
5.1 Relevance Feedback and Pseudo Feedback.....	29
5.2 Experimental Setup and Data.....	30
5.2.1 Experimental Setup.....	30
5.2.2 Experimental Data .....	31
5.2.3 Evaluation Measure .....	31
5.3 Relevance Feedback Experiments .....	31
5.3.1 Pseudo Relevance Feedback.....	33
5.3.2 Relevance Feedback using Residual Collection .....	34
5.3.3 Relevance Feedback using Residual Collection with Overlap in Seen Elements Removed .....	36
5.3.4 Diversity in Retrieved Elements after Removal of Overlap .....	38
<b>Chapter 6. Conclusions and Future Work .....</b>	<b>40</b>
6.1 Conclusions.....	40
6.2 Suggested Future Work.....	41
Bibliography .....	43

Appendix A: Tables .....	44
Appendix B: Flex Class and Sequence Diagram .....	57

## List of Tables

3.1	Values for Exhaustivity.....	16
3.2	Values for Specificity.....	16
4.1	Slope and Pivot values used in Flexible Retrieval.....	24
A.1	P@20 for Pseudo Feedback and M=20.....	44
A.2	P@20 for Pseudo Feedback and M=10.....	45
A.3	P@20 for Pseudo Feedback and M=5.....	46
A.4	P@20 for Relevance Feedback using Residual Collection and M=20.....	47
A.5	P@20 for Relevance Feedback using Residual Collection and M=10.....	48
A.6	P@20 for Relevance Feedback using Residual Collection and M=5.....	49
A.7	P@20 for Relevance Feedback using Residual Collection and with Overlap in Seen Elements Removed, for M=20.....	50
A.8	Number of elements of each type in the basic retrieval.....	51
A.9	Number of elements of each type in feedback retrieval for num_expand = 5.....	52
A.10	Number of elements of each type in feedback retrieval for num_expand = 10.....	53
A.11	Number of elements of each type, in feedback retrieval for num_expand = 25.....	54
A.12	Number of elements of each type in feedback retrieval for num_expand = 50.....	55
A.13	Number of elements of each type in feedback retrieval for num_expand = 500.....	56

## List of Figures

2.1	Rocchio's Algorithm for Relevance Feedback.....	9
3.1	An XML formatted IEEE article.....	11
3.2	An example of a CO Query.....	12
3.3	An example of a CAS Query.....	12
4.1	Format of physical structure of document.....	18
4.2	Pivoted Normalization.....	20
4.3	Flexible Retrieval System.....	24

# Chapter 1

## Introduction

Information retrieval is the science of searching a large store of unstructured data to satisfy the information need of the user. Information retrieval systems like Google and Yahoo Search are used by millions of people everyday to search for information on the Internet. Over the years, computer users have generated a huge amount of data in form of documents, reports, technical and informative articles, email, web-pages, online catalogues and libraries, etc. We need efficient techniques to search through this extensive data and retrieve relevant and useful information when needed. The science of information retrieval deals with different techniques and methods to achieve this goal. The discipline of information retrieval is a broad one and it often references techniques from other disciplines such as statistics, natural language processing and machine learning. Information retrieval is the art of choosing the right retrieval technique and tuning it so that it returns the most useful information on request.

The data to be searched may be structured or unstructured. Structured data have a well-defined organization (for example, a dictionary), while unstructured data have very little or no organization (for example, the body of an email). A lot of data found online in web-pages is highly unstructured. But many online libraries, data repositories and technical publications are increasingly using a format called Extensible Markup Language (XML) to organize their data. XML is a hierarchical organization of data

where documents are composed of data units called entities. Each entity in turn can contain zero or more data entities. Every entity is marked with a label, called the XML tag. If a technical article were organized as an XML document then the parent entity would be labeled by an XML tag, say <article>, which would itself contain several entities labeled by XML tags, say <section>. A <section> might further contain several entities labeled by XML tags, say <sub-section>, and each sub-section would contain several paragraphs labeled by XML tags <paragraph>. XML provides a robust yet simple text format that is widely used for large-scale electronic publications.

The increasing use of XML to organize data has prompted the development of information retrieval techniques that leverage the XML structure for accurate retrieval. These techniques can exploit the structure of the document to produce better retrieval results. In a traditional information retrieval system, the unit of retrieval is the entire document. However, a retrieval system for structured documents can return any ‘part’ of the document in response to the user’s query. The retrieval unit is any data item from the document. For example, a certain section of a document, an image from a web-page, a table from a technical article or a code segment from a tutorial might be returned to the user as a search result.

XML-based information retrieval techniques target documents organized in XML format. These make use of not only the content but also the logical structure of the document during retrieval. What is returned in response to a query depends not only on content but also on the location of the data item in the document. The techniques used in

an XML-based information retrieval system can be extended to handle any document store containing documents with minimal structure. For the purpose of this thesis however, we consider only XML documents.

This thesis focuses on (1) developing an information retrieval system for XML documents which retrieves the most relevant part(s) of a document in response to a query (*flexible retrieval*) and (2) using this system, applying *relevance feedback* to improve the results. Relevance feedback is a mechanism by which information from the relevant and non-relevant results of one retrieval is used to modify the query in such a way that more relevant elements can be retrieved when this modified query is submitted to the system.

The University of Minnesota Duluth (UMD) is a participant in INEX (Initiative for Evaluation of XML Retrieval). [1] This initiative encourages the development of XML-based information retrieval systems. The work done in this thesis pertains to the task described in the ad-hoc retrieval and relevance feedback tracks of INEX. INEX provides the participant organizations with an experimental test collection and uniform scoring mechanisms to evaluate their XML-based retrieval systems.

In the following sections, we discuss the implementation of our flexible retrieval system and the application of relevance feedback to improve the results of this system. Chapter 2 provides a basic introduction to information retrieval, retrieval models and relevance feedback. Chapter 3 describes the data used for experimentation and the evaluation measures. Chapter 4 explains the design and implementation of the flexible retrieval system developed at UMD. In Chapter 5 we discuss the relevance feedback

experiments and their results. The Appendix tabulates the results. We conclude with Chapter 6 with a brief discussion of suggestions for future work.

## Chapter 2

### Basic Concepts of Information Retrieval

Information retrieval is the study of strategies to represent and organize data and retrieve useful information from it. It aims at providing a user with relevant information in response to his information need. The information need of a user is known as a *query*. Satisfying a user's query is more than just doing a keyword match of his query with the documents in the data repository. Information retrieval goes beyond this and provides the user with *information* on the subject of his query. It does not bind to a set of well-defined rules to search for relevant documents. It may return documents that have the same subject as the query but do not contain any words from it.

#### 2.1 Indexing, Weighting and Retrieval

Information retrieval broadly deals with techniques to represent data and algorithms to retrieve information. The internal representation of document data is known as *indexing*. Indexing is the conversion of this data to a format that is easily searchable. One popular internal representation is the *inverted index* where the document collection is converted to a set of keywords along with a corresponding listing of the documents in which the keyword appears. A keyword can be *stemmed* (reduced to its root form). Stemming helps decrease the number of keywords that are morphologically similar. Words that occur so frequently that they do not carry any discriminatory significance (i.e., stopwords) may be ignored during keyword extraction.

Some keywords may be more representative of the subject matter of the document. To account for this, each keyword can be given a *weight* that indicates its importance in the document. Thus the inverted index representation now contains a set of stemmed keywords, along with a list of the documents containing that keyword and its weight. The weight of a term in a document may simply be the number of times it appears in the document. The higher the frequency, the greater its importance. This is known as term frequency (tf) weighting. The number of documents a term appears in can also be used as a factor in weighting. The more documents that a term appears in, the less its discriminatory power for those documents. This is known as inverse document frequency (idf). The tf-idf weighting scheme is widely used in text retrieval systems.

Three popular models in information retrieval are the Boolean Retrieval Model, Vector Space Model and Probabilistic Model. Boolean retrieval is based on propositional logic. In the vector space model, documents and queries are represented as vectors in the space of indexed keywords and the correlation of a document to a query is computed by the geometric distance between them. In the probabilistic model, the probability that a document is relevant to the query is computed by making assumptions about the distribution of the keywords in relevant and non-relevant documents in the index. We use the Vector Space Model in our flexible retrieval system. The following sections give a brief overview of this model and its extensions.

## **2.2 Vector Space Model**

In the Vector Space Model, proposed by Salton [2], each document and query is represented as a weighted vector of terms. The relationship of a document to a query is determined by the distance between the two vectors in the term vector space. The closer the two vectors, the more potentially relevant the document is. Cosine similarity is one of the measures used to compute this distance. It measures the cosine of the angle between the two vectors.

## **2.3 Extended Vector Space Model**

Fox proposed an extension to the traditional vector space model in 1983, which he called the Extended Vector Space Model. [3] This model provides the capability to add document structure information to the document vectors. Thus a document vector will now contain identifiers for both the content and the structure of the document. In the Extended Vector Space Model, a document vector consists of a set of sub-vectors, with each sub-vector representing one class of information in the document. For example, a document may contain different fields (e.g., author name, date of publication, abstract, keywords, main body, and bibliography). Extracting keywords from all these fields and putting them together in one term vector causes loss of structure information. Instead, a term vector can be formed for each class of information and identified by the class type. The document vector will now contain all these vectors, along with their class type information. The similarity between two vectors is now a linear combination of the

similarity between their corresponding sub-vectors. This extension allows for class-based weighting and retrieval.

## **2.4 Smart Retrieval Engine**

Our flexible retrieval system uses Smart 13.0 [4] as its retrieval engine. Smart is an implementation of Salton's Vector Space Model and is a text indexing and retrieval system subsequently developed by Buckley and others at Cornell. It supports Fox's Extended Vector Space Model. Smart consists of software routines to index a document collection, perform retrieval on it and evaluate the results. The Smart index typically contains an inverted file, document vectors, dictionary, and a file containing location information of the indexed documents. Smart also provides a set of advanced term weighting schemes. Evaluation routines can be used to evaluate the retrieval results of various experimental runs.

## **2.5 Relevance Feedback**

Relevance feedback is the technique by which information from relevant and non-relevant documents identified in one retrieval is used to improve the retrieval results in the next. The original query is modified by adding or increasing the weight of terms from relevant documents and/or decreasing the weight of terms from non-relevant documents so as to retrieve more relevant documents. One popular relevance feedback algorithm is Rocchio's, which is stated as:

$$Q' = \alpha * Q + \beta * (D_p/N_p) + \gamma * (D_n/N_n)$$

where, Q': Modified Query    Q: Original Query

$\alpha$ : Weight given to terms from the original query

$\beta$ : Weight given to terms from relevant documents

$\gamma$ : Weight given to terms from non-relevant documents

D<sub>p</sub>: Terms from relevant documents, N<sub>p</sub>: Number of relevant documents

D<sub>n</sub>: Terms from non-relevant documents, N<sub>n</sub>: Number of non-relevant documents

Fig 2.1 Rocchio's Algorithm for Relevance Feedback [5]

We use Rocchio's algorithm for relevance feedback in our flexible retrieval system.

## Chapter 3

### Experimental Data and Evaluation Measures

The data used in our experiments has been generated through INEX. INEX also provides evaluation procedures to measure the performance of XML retrieval systems. As participants at INEX in 2002-2004, we contributed to the development of queries and their relevance judgments. The experimental data provided by INEX consist of a document collection, a set of queries or topics, and relevance judgments for these queries.

#### 3.1 Document Collection

The INEX 2004 document collection is a set of full-text documents formatted using XML. It is a set of 12,107 articles of the IEEE Computer Society's publications from 12 magazines and 6 transactions, covering a period of 1995-2002. It is approximately 500MB in size. The collection has a suitably complex XML structure and contains articles of different lengths.

A typical XML document contains front-matter, body and back-matter. The front-matter contains information like the title, date of publication, author name, abstract, etc. The body contains sections, sub-sections, paragraphs, tables, lists, figures, etc. The back-matter contains acknowledgements, bibliography, etc. The document follows a DTD. Figure 3.1 is an example of a document from the INEX collection.

#### 3.2 Query Collection

INEX queries are of two types: *Content Oriented (CO)* and *Content and Structure (CAS)*.

```

<article>
  <fno> A1061 </fno>
  <doi> 10.1041/A1061s-1995 </doi>
  <fm>
    <ti> IEEE Annals of the History of Computing </ti>
    <pdt>
      <mo> Spring </mo>
      <yr> 1995 </yr>
    </pdt>
    <au sequence="first">
      <fnm> James </fnm>
      <snm> Tomayko </snm>
      <role> Editor </role>
    </au>
    <edinfo>
      <p> Editor's note </p>
    </edinfo>
    <abs>
      <p> Abstract </p>
    </abs>
  </fm>
  <bdy>
    <sec>
      <ss1>
        <st> Section Title </st>
        <p> Paragraph 1 </p>
        <p> Paragraph 2 </p>
      </ss1>
    </sec>
    <sec>
      <p> Paragraph 1 </p>
    </sec>
  </bdy>
  <bm>
    <bib>
      <bibl>
        <bb>
          <au> Author name of author in bibliography </au>
        </bb>
      </bibl>
    </bibr>
  </bm>
</article>

```

Figure 3.1 An XML formatted IEEE article

These queries are developed by the participants of INEX every year. The CO queries are free-text queries that ignore the document structure and stress the *content* of the document. CAS queries contain explicit *structural constraints*. They specify which component of the document the relevant terms should come from and which components

should be returned. There are 34 CO queries and 40 CAS queries in the INEX 2004 collection. The following figures show an example of a CO query and a CAS query respectively.

```
<inex_topic query_type="CO+S" ct_no="42">
<InitialTopicStatement>Ray tracing is a concept used to render surfaces of 3-d
objects in a 2-d plane. I want to know how ray tracing is used to achieve the same
for images. Any material that talks about the use of ray tracing in images would be
treated as useful. <InitialTopicStatement>
<title>+"ray tracing" images</title>
<description>Return articles that talk about the technique of Ray Tracing used in
images. </description>
<narrative>Relevant articles should talk about ray tracing and how this concept is
used to render surfaces of 3-D objects in a 2-D image... </narrative>
</inex_topic>
```

Fig 3.2 An example of a CO Query [6]

```
<inex_topic query_type="CAS" ct_no="61">
<InitialTopicStatement>To Find Information regarding data embedding using
watermarking.<InitialTopicStatement>
<castitle>//article[about(//p,"data
embedding")]//p[about(.,watermarking)]</castitle>
<description>We are looking for paragraphs describing watermarking in articles
which describe data embedding.</description>
<narrative>In today's world the issue of data security is highly significant.One such
technique to ensure data security is steganography... </narrative>
</inex_topic>
```

Fig 3.3 An example of a CAS Query [6]

### 3.3 Relevance Judgments

For the purpose of evaluation INEX provides a set of relevance assessments for every query. These relevance assessments are contributed by the participants. After queries are developed and submitted, each participating organization uses its system to retrieve results for these queries. These results are grouped by query and distributed equally

among all participants. Each group uses the results retrieved for the query assigned to them and manually assesses every component of those documents. A relevance score is given to each component and the document as a whole. The score is composed of two measures:

- *Exhaustivity (e-value)*: Describes the extent to which the document element discusses the topic. [7]
- *Specificity (s-value)*: Describes the extent to which the document element focuses on the topic. [7]

Exhaustivity and Specificity have 4 possible values.

e-value	Meaning
0	not exhaustive: the component does not discuss the query
1	marginally exhaustive: the component discusses some aspects of the query
2	fairly exhaustive: the component discusses many aspects of the query
3	very exhaustive: the component discusses most or all aspects of the query

Table 3.1 Values for Exhaustivity

s-value	Meaning
0	not specific: the query subject is not the theme of the component
1	marginally specific: the query subject is a minor theme of the component
2	fairly specific: the query subject is a major theme of the component
3	very specific: the query subject is the only theme of the component

Table 3.2 Values for Specificity

A document component may be assessed with any combination of e-value and s-value, with (3,3) being highly exhaustive and highly specific and (0,0) being completely non-relevant.

### 3.4 Evaluation Measures

INEX provides the participants with uniform evaluation metrics. Two measures have traditionally been used to evaluate the performance of the retrieval system:

- *Precision*: This is the ratio of the number of relevant documents retrieved to the total number of documents (relevant as well as non-relevant) retrieved.
- *Recall*: This is the ratio of the number of relevant documents retrieved to the number of relevant documents in the index.

Recall and Precision are inversely related. If all the documents in the index are retrieved, recall is 100% but precision will be very low. One way in which the retrieval system is evaluated is by measuring its precision at different values of recall and then taking an average of this accumulated precision (avg P). The higher the average precision, the better is the performance.

To apply the recall-precision metrics, the two dimensions of relevance need to be combined into a single relevance score for a document component. This is done by using a quantization function. The following quantization functions are used in INEX 2004 evaluation metrics: [8]

- *Strict Quantization*: This function considers only highly exhaustive and highly specific components as relevant.

$$f_{strict}(e, s) := \begin{cases} 1 & \text{if } e = 3 \text{ and } s = 3, \\ 0 & \text{otherwise.} \end{cases}$$

- *Generalized Quantization*: This function credits components according to their degree of relevance.

$$\mathbf{f}_{gen}(e, s) := \begin{cases} 1 & \text{if } (e, s) = (3, 3), \\ 0.75 & \text{if } (e, s) \in \{(2, 3), (3, \{2, 1\})\}, \\ 0.5 & \text{if } (e, s) \in \{(1, 3), (2, \{2, 1\})\}, \\ 0.25 & \text{if } (e, s) \in \{(1, 2), (1, 1)\}, \\ 0 & \text{if } (e, s) = (0, 0). \end{cases}$$

- *Specificity oriented strict quantization*: This is a set of functions that apply strict quantization with respect to specificity and allow different degrees of exhaustivity.

$$\mathbf{f}_{s3_e321}(e, s) := \begin{cases} 1 & \text{if } e \in \{3, 2, 1\} \text{ and } s = 3, \\ 0 & \text{otherwise.} \end{cases}$$

$$\mathbf{f}_{s3_e32}(e, s) := \begin{cases} 1 & \text{if } e \in \{3, 2\} \text{ and } s = 3, \\ 0 & \text{otherwise.} \end{cases}$$

- *Exhaustivity oriented strict quantization*: This is a set of functions that apply strict quantization with respect to exhaustivity and allow different degrees of specificity.

$$\mathbf{f}_{e3_s321}(e, s) := \begin{cases} 1 & \text{if } e = 3 \text{ and } s \in \{3, 2, 1\}, \\ 0 & \text{otherwise.} \end{cases}$$

$$\mathbf{f}_{e3_s32}(e, s) := \begin{cases} 1 & \text{if } e = 3 \text{ and } s \in \{3, 2\}, \\ 0 & \text{otherwise.} \end{cases}$$

## **Chapter 4**

### **Flexible Retrieval System**

#### **4.1 What is Flexible Retrieval**

Flexible Retrieval exploits the document structure to decide the granularity of retrieval. Given a query, this approach extracts the highly correlating part(s) of the document, be it a paragraph, section or the document as a whole. Flexible retrieval is a more focused approach to search. It is based on the idea that certain parts of a document may be more representative of its context than others and that certain parts may be more relevant to the query than others. The wide-scale use of XML as a document format encourages the development of flexible retrieval systems.

#### **4.2 Overview of Flexible Retrieval System**

A retrieval system has two important aspects: an internal representation of documents and a retrieval strategy over the document collection. The experimental document collection used by our flexible retrieval system is a set of XML-formatted IEEE articles. Each document is first parsed and broken into its smallest content-bearing elements. In the case of our experimental collection, the content-bearing elements are the leaf elements of the XML document tree, like the paragraph, titles, figure captions, etc. The text of these leaf elements is stemmed and stopwords are eliminated. These ‘stemmed’ and ‘stopped’ leaf elements are then indexed.

For a given a query, a set of rank-ordered highly related leaf elements are retrieved from the index. A retrieved leaf element indicates that the document it belongs to contains some information in common with the query. The flexible retrieval system takes a set of retrieved leaf elements, generates a set of internal (parent) elements and calculates the correlation between the query and all such elements in the parent documents. It then returns a list of elements ranked by correlation.

### **4.3 Preprocessing and Indexing Documents**

Our flexible retrieval system uses Smart as its indexing and retrieval engine. Before indexing, the XML documents are parsed and their content-bearing leaf elements extracted. Each of these elements is then converted into a Smart indexable unit. Apart from its text, each indexable unit contains the following information: abstract, title, editor's information, bibliography, acknowledgements and keywords of the document it belongs to. These fields are added for the purpose of structure-based retrieval, as in the case of CAS queries.

During parsing, the physical structure of the document is also extracted and added to a *document schema repository*. This will be used when the document is rebuilt during flexible retrieval. Figure 4.1 shows the format in which the physical structure is stored. The first field indicates the document element name and its location in the document tree. The second field indicates the number of children it has. The third field indicates whether or not the element has a right sibling. For example, the article above has 5 sub-elements, 4 of which are shown in the example: fno, doi, fm and bdy. The fm has 3 sub-elements:

tig, au and abs, and has a right sibling bdy. This information is enough to rebuild the document tree.

```
/article[1] 5 0
/article[1]/fno[1] 0 1
/article[1]/doi[1] 0 1
/article[1]/fm[1] 3 1
/article[1]/fm[1]/tig[1] 2 1
/article[1]/fm[1]/tig[1]/atl[1] 0 1
/article[1]/fm[1]/tig[1]/pn[1] 0 0
/article[1]/fm[1]/au[1] 2 1
/article[1]/fm[1]/au[1]/fnm[1] 0 1
/article[1]/fm[1]/au[1]/snm[1] 1 0
/article[1]/fm[1]/au[1]/snm[1]/ref[1] 0 0
/article[1]/fm[1]/abs[1] 0 0
/article[1]/bdy[1] 18 1
/article[1]/bdy[1]/sec[1] 3 1
/article[1]/bdy[1]/sec[1]/st[1] 0 1
/article[1]/bdy[1]/sec[1]/p[1] 0 1
/article[1]/bdy[1]/sec[1]/p[2] 0 0
```

Fig 4.1 Format of physical structure of document

After extracting the text and structure, each indexable unit (which can be considered as a small document) is indexed using Smart's routines. The index contains an inverted list of document vectors weighted by term frequency, a dictionary, and a file named '*textloc*' containing location information for the indexed documents. This file maps the indexed "documents" to Smart's document identifiers.

#### 4.4 Retrieving and Scoring Elements

As described, retrieval in the flexible retrieval system consists of two steps: (1) retrieve a list of highly correlating leaf elements and (2) score all elements of a document whose leaf element appears in this list.

#### 4.4.1 Retrieval of Leaf Elements

Each query and document is represented as a weighted vector of terms. We choose the *Lnu-ltu* weighting scheme for these experiments.

##### 4.4.1.1 Weighting Schemes

*Lnu* and *ltu* are variations of the tf-idf weighting scheme. The first letter (L, l) represents the *tf* component, the second letter (n, t) the *idf* component and the third letter (u) the type of normalization used. Details are seen below

(1) *L* uses the following variation on term-frequency:

$$\text{new\_tf}(t) = \frac{1 + \log(\text{tf}(t))}{1 + \log(\text{avg\_tf})}$$

where  $\text{tf}(t)$  is the frequency of term  $t$  and  $\text{avg\_tf}$  is the average term frequency in the document.

(2) *n* (*none*) indicates that no idf weighting is used.

(3) *u* stands for the normalization scheme called *unique normalization* [9]. Term weights are normalized so that longer documents would not have an unfair advantage over shorter documents with respect to retrieval. Normalization reduces all term weights to the same scale. The formula [9] used in unique normalization is

$$(1 - \text{slope}) + (\text{slope}) * (\text{num\_of\_unique\_terms\_in\_doc})/\text{pivot}$$

Under cosine normalization, when the probability of retrieval and the probability of relevance are plotted against document length, the curves appear as shown below [9]:

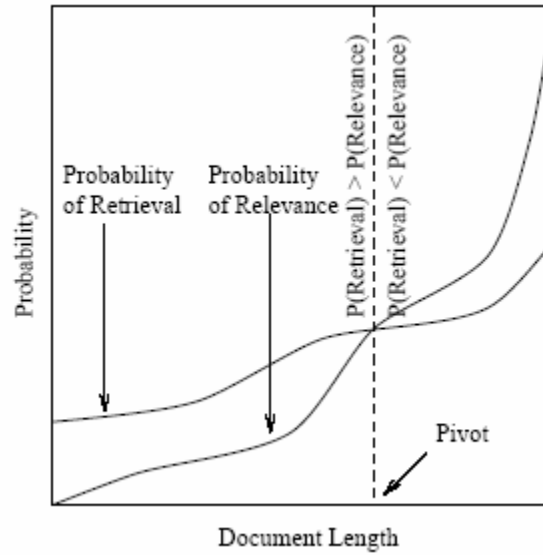


Fig 4.2 Pivoted Normalization. Source: *Pivoted Document Normalization Length Normalization* by Singal, Buckley and Mitra,

*Pivot* is a document length at which the probability of retrieval of a document is the same as its probability of relevance. Documents with length less than pivot have a higher probability of retrieval than relevance and documents with length more than pivot have lower probability of retrieval than relevance. We want to increase the normalization factor for documents that are shorter than the pivot (thus reducing their chances of being retrieved) and decrease the normalization factor for documents that are longer than the pivot (thus increasing their chances of being retrieved). This is controlled by a parameter called *slope*. For our flexible retrieval system we use the empirically generated values for slope and pivot shown in Table 4.1. We use different slope and pivot values for each element type. That is because the size of each element type differs significantly.

Element Type	Pivot	Slope
Paragraph	24	0.06
Sub-section	92	0.01
Section	154	0.1
Article	538	0.03

Table 4.1 Slope and Pivot values used in Flexible Retrieval

(4) In the case of *ltu*, the query weighting scheme *l* uses the following variation of term frequency:

$$\text{new\_tf}(t) = \log(\text{tf}(t)) + 1.0$$

(5) *t* uses the following as the idf factor to scale the term frequency with:

$$\log(\text{num\_of\_docs}/\text{num\_of\_docs\_term\_appears\_in})$$

To summarize, in *Lnu* the weight of a term is computed as:

$$\text{wt}(t) = \frac{1 + \log(\text{tf}(t))}{1 + \log(\text{avg\_tf})} \\ \frac{1 + \log(\text{tf}(t))}{(1 - \text{slope}) + (\text{slope}) * (\text{num\_of\_unique\_terms\_in\_doc})/\text{pivot}}$$

In *ltu* the weight of a term is computed as:

$$\text{wt}(t) = \frac{\log(\text{tf}(t) + 1.0) * \log(\text{num\_of\_docs}/\text{num\_of\_docs\_term\_appears\_in})}{(1 - \text{slope}) + (\text{slope}) * (\text{num\_of\_unique\_terms\_in\_doc})/\text{pivot}}$$

#### 4.4.1.2 Similarity Computation

The similarity of a document to a query can be calculated using cosine or inner product. We use inner product, since the document and query vectors are already normalized. The inner product between two vectors is calculated by:

$$d(D, Q) = \sum (wt(d_i) * wt(q_i))$$

where D and Q are document and query vectors respectively,  $d_i$  is the  $i^{\text{th}}$  term from D and  $q_i$  is the  $i^{\text{th}}$  term from Q belonging to the same class type. The weight of a term is scaled by the weight of the class to which it belongs. Terms in certain fields of the document may be more important than those in other fields. For example, a term in the title may be more indicative of the document's contents than a term in a paragraph.

Recall that a list of leaf elements is retrieved for a given query. The next step is to *populate* each document with a leaf element in this list by building the internal elements of the document and giving scores to them as indicated in the following section.

#### 4.4.2 Populating the Documents

The following steps are used to *populate* a document:

For every retrieved leaf element:

1. Extract the structure of the document from the document schema repository.
2. Populate the leaf elements by fetching the corresponding vectors from the index.

Calculate the similarity between the query and this vector.

3. Move up the tree and populate every non-leaf element as follows:
  - a. Get the term-frequency-weighted term vectors of the child elements.
  - b. Merge the vectors into a single vector. Term frequencies of common terms are added together. This new vector is the term-frequency-weighted term vector for this element.
  - c. Re-weight the vector using *Lnu* weighting. Use appropriate values of slope and pivot depending upon the type of the element.
  - d. Compute the similarity between this document vector and the query vector using inner product. Store this similarity value for the element.
4. Add the elements to a list sorted by similarity.

Once all documents are populated and their elements added to a list sorted by similarity, the top N elements are returned as a response to the query.

## **4.5 Design of the Flexible Retrieval System**

The flexible retrieval system consists of 4 modules:

1. Input processing
2. Flex – the flexible retrieval system core
3. Smart Interface.
4. Output processing

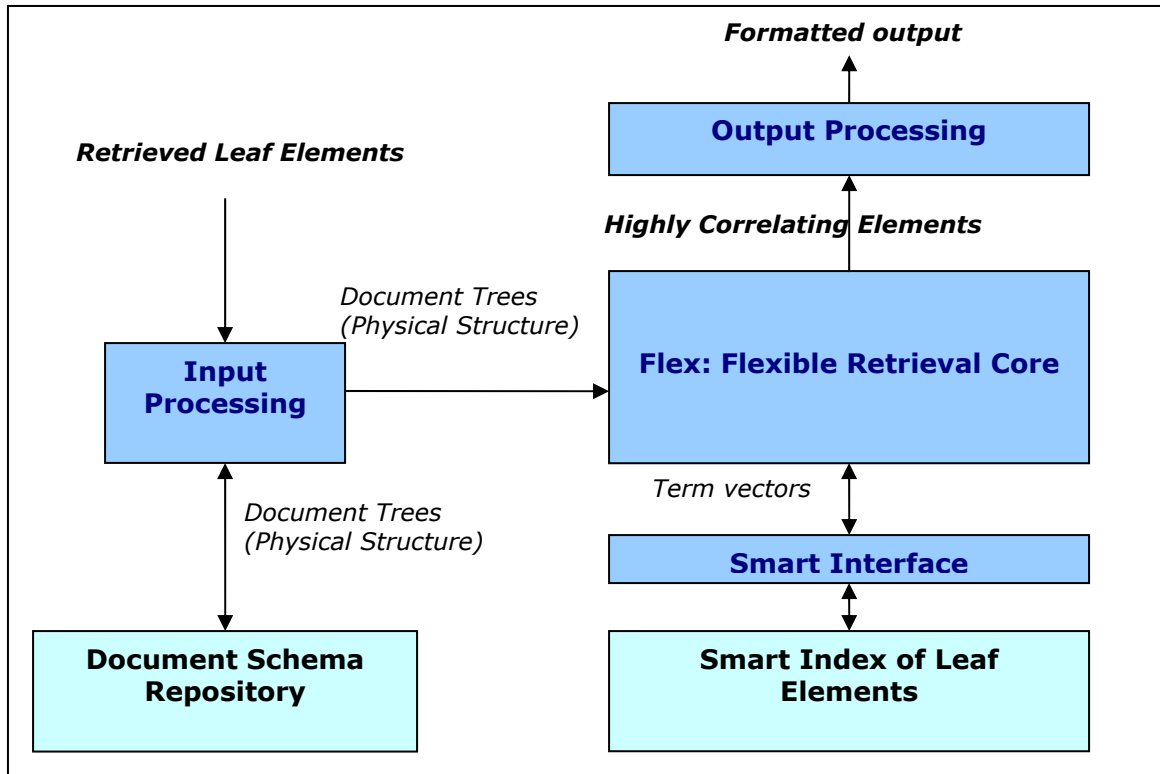


Fig 4.3 Flexible Retrieval System

#### 4.5.1 Input Processing

The input processing module takes the list of retrieved leaf elements for a query and extracts the physical structure of the parent documents of these elements from the document schema repository. It then records the Smart identifier and relevance score, if present, for every leaf element against its entry in the document structure. This will be used by Flex.

#### 4.5.2 Flex – the Flexible Retrieval Core

The Flex system is very modular and extensible. It consists of 4 classes:

1. **Properties:** This class is responsible for the configuration properties of Flex. It reads the properties from a configuration file and provides them to Flex. The following properties are used:
  - a. *Weighting scheme:* This indicates what weighting scheme will be used to weight the vectors. Currently, there are 2 possibilities: *Lnu* and *Avg*. *Avg* simply takes the average of the term weights while merging vectors to make a parent vector.
  - b. *Similarity scheme:* This indicates what scheme will be used to compute similarity. Currently, the 2 possibilities are inner product and cosine.
  - c. *Number of results:* This is a number indicating how many elements to return to the user.
  - d. *Size Threshold:* This is a value indicating the minimum size (number of terms) of the elements to return.
  - e. *Similarity Threshold:* This is a value indicating the minimum similarity value an element to be returned should have.
  - f. *Slope and Pivot:* Slope and Pivot values can be set for different element types (paragraph, sub-section, section and article), for unique normalization.



- f. *Number of children*: The number of children it has in the document tree
- g. *Pointers to parent, leftmost child and right sibling*.

### **4.5.3 Smart Interface**

This module interfaces the flexible retrieval system with Smart. The flexible retrieval system accesses Smart's index using this module. Access to the index is required for reading the term vectors of leaf elements and query.

### **4.5.4 Output Processing**

The output processing unit formats the system results. The results are formatted for:

1. Submission to INEX for evaluation: INEX requires results to be submitted in a certain XML format.
2. Experimental analysis: We have developed our own modules for comparing results with the relevance assessments.
3. Smart evaluation: Smart also provides an evaluation module that requires results to be submitted in a certain format.

## **4.6 Implementation of the Flexible Retrieval System**

The flexible retrieval system has been implemented mainly in C++ and Perl. The modules that process the input and output data are written in Perl, while C++ is used for the core flexible retrieval algorithm. The lightweight Smart interface module has been written in C. In this section we discuss the implementation of Flex class only. The Flex class implements the core flexible retrieval logic. It consists of the following important modules:

1. *Populate*: This module populates a document tree. It takes a document tree and traverses it bottom-up and breadth-first, starting at the leaf elements. It uses the Smart identifier present in the leaf element and its siblings to get their frequency weighted term vectors from the Smart index. It then merges these term vectors to form the parent's term vector, using the *Merge* module. Then, using the *Weight* module, it converts the term vector using the weighting scheme specified by the configuration properties. Finally, it computes the similarity between this vector and the query vector and records that as the (relevance) score for that parent element.
2. *Merge*: This module merges a set of term vectors to form one single vector. The weight of a term common to two or more vectors in the merged vector, is simply the addition of its weights in the individual vectors.
3. *Weight*: This module converts the frequency-based weights of terms in a vector to the weights specified by the configuration properties. Currently, it can be *Lnu* or *Avg*. It takes a term vector, applies the weighting scheme to the term weights and returns the vector.

The flexible retrieval system has been designed in such a way that new modules for input processing, weighting, merging, etc. can be simply plugged in without disturbing the classes that implement the core system.

## Chapter 5

### Relevance Feedback

#### 5.1 Relevance Feedback and Pseudo Feedback

Relevance feedback is a technique where the search system tries to retrieve documents that are “more like” the relevant documents retrieved the first time. An initial retrieval is performed and the results assessed. It uses information from the previous retrieval to achieve this. The query is modified by adding terms from relevant documents and re-weighting the original query terms that occurred in relevant and/or non-relevant documents in an attempt to retrieve more relevant documents the second time.

In *relevance feedback*, the user specifies which documents are relevant and which are not, from the result of the first retrieval. In *pseudo feedback*, the retrieval system assumes the retrieved documents to be relevant. It does not require user assistance.

Furthermore, to evaluate relevance feedback the documents used for feedback may not be considered during the feedback retrieval. This is based on the assumption that the user has already seen them and now wants other documents that are as or more relevant than these. This is known as evaluation based on *residual* collection. The other possibility is to *freeze* the seen documents at their original ranks and bring in new, similar documents at other ranks. This is termed as *rank freezing*.

We explored the possibility of applying relevance feedback to the results achieved through our flexible retrieval system. In this chapter we discuss the experiments performed and their results.

## **5.2 Experimental Setup and Data**

### **5.2.1 Experimental Setup**

Before implementing relevance feedback in our flexible retrieval system, we first experimented with it on a system that imitates the behavior of our flexible retrieval system. This was done to verify the effectiveness of relevance feedback and determine which parameters produce the best results.

To build a retrieval system that imitates flexible retrieval, we first prepared an index containing all the elements of a document. As described, the flexible retrieval system indexes only the content-bearing leaf elements of a document and uses these to build the text of their parents and score them for a given query. Our experimental system indexes all the elements of a document beforehand. A Smart retrieval run against this index then returns a rank-ordered list of highly correlating elements. This list should theoretically be an exact match to the list retrieved by our flexible retrieval system. After experimentation, we have verified that both the systems return the same elements in response to a given query, with minor differences in the rank-ordering.

This approach for our experiments lets us use Smart's capabilities for relevance feedback. Once we determine the parameters that produce the best results with this

experimental system, we can use them to implement relevance feedback in our flexible retrieval system.

### **5.2.2 Experimental Data**

Relevance assessments form a very important part of our experiments. They are used for (a) indicating which of the documents being used for feedback are relevant and which are not and (b) evaluating the performance of relevance feedback. We first process the assessments provided by INEX and create a set of relevant elements based on their exhaustivity and specificity values. An element with e-value 1, 2 or 3 and s-value 1, 2 or 3 is chosen as relevant. There are many ways in which relevant elements can be chosen. A set of elements with e-value=3 and s-value=3 will be a ‘highly relevant highly specific’ set, a set with e-value=3 and s-value=1, 2 or 3 will be a ‘highly relevant’ set, and so on. We choose the set with any non-zero e-value and s-value for an element because it’s the most exhaustive set and a good starting point for experimentation.

### **5.2.3 Evaluation Measure**

$P@n$ , precision value after retrieval of  $n$  elements, is used to measure the effectiveness of relevance feedback. This precision value is compared against the  $P@n$  for the basic (initial) retrieval run. The value of  $n$  used in most cases is 20.

## **5.3 Relevance Feedback Experiments**

We experimented with the following types of relevance feedback:

1. Pseudo Feedback
2. Relevance Feedback using Residual Collection

3. Relevance Feedback using Residual Collection after removing *overlap* from documents used for feedback.

We use a sub-set of the retrieved elements for feedback, called the *seen\_docs* set. This set may contain *overlapping* elements. Overlap refers to the presence of a descendant of a document element in the *seen\_docs*, along with the element. Whether overlapping elements should be used for relevance feedback is an important consideration. The first two sets of experiments allow overlap in *seen\_docs*, while the third set does not. Not only are these *seen\_docs* removed from consideration during the feedback retrieval, but also their descendants are removed from the retrieved list.

We used Rocchio's algorithm for relevance feedback. Smart provides an implementation of this algorithm. We experimented with a wide range of values for the following parameters:

1. Number of documents to be used for relevance feedback, called *seen\_docs*
2. Number of terms to be added to the modified query, called *num\_expand*.
3. Weights given to original, relevant and non-relevant components, called  $\alpha$ ,  $\beta$  and  $\gamma$ , respectively.

The following values were used:

1. *seen\_docs*: 20, 10, 5
2. *num\_expand*: 5, 10, 25, 50, 100, 300, 500
3.  $\alpha$ ,  $\beta$  and  $\gamma$ :

1, 1, 0	2, 1, 0	3, 1, 0	4, 1, 0	5, 1, 0
1, 2, 0	2, 3, 0	3, 2, 0	4, 3, 0	5, 2, 0
1, 3, 0	2, 5, 0	3, 4, 0	4, 5, 0	5, 3, 0
1, 4, 0		3, 5, 0		5, 4, 0
1, 5, 0				

### **5.3.1 Pseudo Relevance Feedback**

#### **5.3.1.1 Approach**

1. Consider top M elements, from the retrieved N as *seen\_docs*.  $M < N$ .
2. Assume them to be relevant and use them to modify the query.
3. Retrieve N elements.

#### **5.3.1.2 Results**

Please refer to tables A.1 to A.3 in Appendix A.

#### **5.3.1.3 Observations**

1. Percentage of improvement at P@20 is largely dependent on the values of alpha and beta, for smaller values of M (5 to 10).
2. Percentage of improvement at P@20 is also dependent on the number of terms added to the query (num\_expand). This dependence is more prominent in the case of M=20 where improvements can be seen only after at least 100 terms are added.

#### **5.3.1.4 Conclusions**

1. Number of terms added: Pseudo Feedback starts showing improvements in results when a large number of terms are added to the query. When the number of seen documents is higher (20), this improvement is observed for any value of alpha and beta. When the number of seen documents starts decreasing, alpha and beta have a greater influence on relevance feedback.
2. Alpha and Beta Values: Weighting terms from original documents (alpha) 3 to 5 times more than terms from seen documents (beta) gives better results. This may

be because in pseudo feedback we *assume* the seen documents to be relevant while in reality they may or may not be. Thus letting the original terms have more influence on the feedback retrieval, with only a little impetus given by the new terms, reduces the risk of tilting the results toward non-relevant documents that were assumed relevant.

### **5.3.2 Relevance Feedback using Residual Collection**

#### **5.3.2.1 Approach**

1. Consider top  $M$  elements, from the retrieved  $N$  as *seen\_docs*.  $M < N$ .
2. Use relevance assessments to indicate which elements are relevant and which are not. Use them to modify the query.
3. Retrieve  $N$  elements removing the  $M$  *seen\_docs* from consideration for retrieval.

#### **5.3.2.1 Results**

Please refer to table A.4 to A.6 in Appendix A.

#### **5.3.2.3 Observations**

1.  $P@20$  for Relevance Feedback using Residual Collection is higher than  $P@20$  for Pseudo Feedback. The best improvement over the base run using Pseudo Feedback is 9% while the best improvement over the base run using Residual Relevance Feedback is 18%.
2.  $P@20$  is dependent on the value of  $M$ . As  $M$  decreases (20 to 5), the  $P@20$  at most of the points in the table increases, even when fewer terms (*num\_expand*) are added to the query.

3.  $P@20$  is dependent on the number of terms added to the query (`num_expand`). It increases with an increase in `num_expand`.
4. Alpha and beta values or the difference between them does not have significant influence on  $P@20$ . For any value of  $M$  (20 to 5), the percentage of improvement over the base run varies only slightly with variation in alpha and beta values.

#### **5.2.3.4 Conclusions**

1. Number of terms added: Relevance Feedback using Residual Collection improves with increase in the number of terms added to the query.
2. Alpha and Beta: Since in Relevance Feedback we know which documents are relevant and which are not, only terms from relevant documents are added to the query (unlike Pseudo Feedback). Thus increasing their weight (beta) does not worsen the result. This may be a reason why the precision is not very sensitive to variations in alpha and beta.
3. One possible explanation why Relevance Feedback performs better for smaller values of  $M$  is that with smaller  $M$  we remove fewer relevant documents from consideration during the feedback run.
4. Relevance Feedback may perform better than Pseudo Feedback because we know exactly which documents are relevant and hence add only good terms to the query.

### **5.3.3 Relevance Feedback using Residual Collection with Overlap in Seen Elements Removed**

In this experiment we remove overlapping elements from the elements used for feedback. Also, during feedback retrieval we remove not only the seen elements from consideration but also all their descendants.

#### **5.3.3.1 Approach**

1. From the  $N$  retrieved elements choose the top  $M$  non-overlapping elements as *seen\_docs*.  $M < N$
2. Use relevance assessments to indicate which elements are relevant and which are not. Use them to modify the query.
3. Retrieve  $N$  elements, removing the  $M$  *seen\_docs* and all their descendants from consideration during retrieval.

#### **5.3.3.2 Results**

Please refer to table A.7 in Appendix A.

#### **5.3.3.3 Observations**

1.  $P@20$  for Relevance Feedback shows improvement over that of the base run only at a few points in the table. The percentage of improvement over the base run is at best 3%.
2. There is no improvement in  $P@20$  until at least 100 terms are added to the query, after which significant improvement is seen only for higher values of alpha, like 4 and 5.

#### 5.3.3.4 Conclusions

1. Number of terms added: A large number of terms need to be added for relevance feedback to perform better than the base run.
2. Alpha and Beta: Improvements are seen when alpha is about 2.5 to 3 times greater than beta.
3. The performance of this type of relevance feedback is not as good as the performance of relevance feedback that allows descendants of seen elements to be retrieved. The relevant content of an element used in feedback comes from its descendants. Removal of seen elements from feedback retrieval (residual collection) negatively affects the result and removal of their descendants further hampers it.
4. However, the improvement seen in some cases points towards the possibility that removal of overlap from the seen elements and removal of their descendants from the feedback result has retrieved elements different from those in the basic retrieval, yet relevant. This is in line with the aim of this experiment. It challenges the monopoly of relevant seen elements by removing their descendants from the retrieved result and letting other elements take their place.

Overlap in the *seen\_docs* tends to retrieve more descendants, siblings or parents of the seen elements after feedback. With the removal of overlap, we hope to get a more diverse set of elements in the result of feedback retrieval. We conducted experiments to verify this. These experiments and their results are explained in the following section.

### 5.3.4 Diversity in Retrieved Elements after Removal of Overlap

In these experiments we analyze the result of basic retrieval along with the relevance feedback retrieval with and without removal of overlap, to determine the diversity of the retrieved elements. We find the average number of distinct documents represented in each result. We then count the number of paragraphs, sub-sections, sections and whole documents in the result. This is used to compute the average number of descendants per parent element in the retrieval result, to get a measure of overlap.

Table A.8 in Appendix A shows the average number of documents, paragraphs, sub-sections and sections present in the basic retrieval, while tables A.9 to A.13 show the same statistics for relevance feedback using residual collection with and without overlap from (and children of) *seen\_docs* removed.

From the tables we can see that the average number of paragraphs per document in the basic retrieval is approximately 0.6. In the case of relevance feedback with overlap (and children of *seen\_docs*) removed, the average number of paragraphs per document is 0.06 (num\_expand at least 25) to 0.2 (num\_expand 5). Overlap is significantly less in the latter case. As expected, diversity is introduced in the retrieval result by removal of overlapping elements in *seen\_docs*. This has a significant impact on relevance feedback in the flexible retrieval environment. Removal of overlap decreases the bias towards retrieving elements related to the seen elements.

In this chapter we discussed the relevance feedback experiments. We propose to use these observations while implementing relevance feedback in our flexible retrieval

system. The next chapter summarizes the conclusions and provides a few suggestions for future work.

## Chapter 6

### Conclusions and Future Work

#### 6.1 Conclusions

The aim of this thesis was to apply relevance feedback in the flexible retrieval environment and study the effect of feedback parameters on the retrieval. We started with a retrieval system that imitated flexible retrieval. We then experimented with different types of relevance feedback, namely, pseudo feedback and relevance feedback using residual collection with and without overlap to determine what combination of feedback parameters improves the retrieval result.

Unlike traditional retrieval systems, where a set of documents is returned in response to a query, our flexible retrieval system returns a set of document elements. This gives rise to the possibility of overlapping elements in the retrieval result. We studied how overlap influences the feedback results.

It was observed that relevance feedback has the potential of improving the results in a flexible retrieval environment. Results improve when a large number of terms are added to the query in the feedback step. In case of pseudo feedback, giving more weight to original terms than to terms added from relevant documents produces better results. In case of relevance feedback the difference between the weights of these two components does not have significant influence, but results are better when original terms are given higher weights. This behavior can be explained by the fact that in pseudo feedback we

*assume* seen elements to be relevant while in relevance feedback we *know* exactly which seen elements are relevant. In pseudo feedback there is a risk of adding terms from elements incorrectly assumed as relevant. Letting the original terms have more influence on retrieval than the new terms reduces this risk.

In relevance feedback using residual collection, results improve as the number of seen elements decreases. A residual collection does not contain the seen elements. We have essentially discarded elements that were highly correlating from the feedback retrieval. Thus, as the number of seen elements decreases, the number of relevant elements being discarded from retrieval also decreases. This is one reason why the performance improves as number of seen elements reduces. This behavior also indicates that it is possible to retrieve a considerable number of relevant elements with relevance feedback even after using only a few elements for feedback.

The experiments also indicate that overlap plays an important role in relevance feedback. Flexible retrieval tends to retrieve elements from the same documents since if an element is relevant, some of its descendants will also be relevant (the content of an element is contained in its descendants). If this overlap is allowed in the elements used for feedback, more elements from the same documents will be retrieved. Though this not undesirable, other relevant documents may be ignored due to this. It can be seen from the experimental results that removal of overlap leads to a more diverse set of elements.

## **6.2 Suggested Future Work**

These experiments provide a good set of relevance feedback experiments in a flexible retrieval environment and provide information on the influence of different feedback

parameters on retrieval. There is still a large scope for other experiments. A few suggestions are listed below.

1. We would like to study the effect of rank freezing on the feedback retrieval results.
2. We need to explore the effect of using relevance assessments with different levels of exhaustivity and specificity of elements. For the experiments here we used elements with any non-zero e-value and s-value as relevance assessments. Experiments can be performed using assessments over a range of (e-values, s-values) such as (3, x), (2, x), (2-3, x), (x, 3), (x, 2), (x, 2-3) where x stands for 1, 2 or 3. Experimental data tells us that the number of relevance assessments starts decreasing as we go from (x, x) to (3, 3). We need to study the effect of using only a few highly relevant and highly specific assessments versus using many fairly relevant and fairly specific assessments.
3. We need to study the effect of overlap in more detail. The initial experiments show that overlap affects retrieval. We need to consider overlap from the context of the user: is retrieving more elements from the same documents more desirable than retrieving elements from different documents? How can the information provided by overlap be used to retrieve better elements in the feedback run?

We can then use the results from this exhaustive set of experiments to implement relevance feedback in our flexible retrieval system.

## Bibliography

- [1] Fuhr, N; Malik, S; Lalmas, M. Overview of the INitiative for the Evaluation of XML retrieval (INEX) 2003. *INEX 2003 Workshop Proceedings*, Schloss Dagstuhl, December 15-27, 2003
- [2] Salton G., Wong A., Yang C. A Vector Space Model for Automatic Indexing. *Comm. ACM* 18(11), 613-620, 1975
- [3] Fox, E. Extending the Boolean and Vector Space Models of Information Retrieval with P-norm Queries and Multiple Concept Types. Ph.D. Dissertation, Department of Computer Science, Cornell University, 1983.
- [4] Salton, G., editor. *The SMART Retrieval System – Experiments in Automatic Document Retrieval*, Prentice-Hall, Englewood Cliffs, NJ, 1971.
- [5] Rocchio, J. Relevance feedback information retrieval. In Gerard Salton (ed.): *The Smart Retrieval System - Experiments in Automatic Document Processing*, 313-323 Prentice-Hall, Englewood Cliffs, N.J., 1971
- [6] INEX, 2004. <http://inex.is.informatik.uni-duisburg.de:2004>
- [7] Kazai, G; Lalmas, M; Piwowarski, B. INEX 2004 Relevance Assessment Guide. *INEX 2004 Workshop Preproceedings*, Schloss Dagstuhl, December 6-8, 2004.
- [8] Vries, A; Kazai G; Lalmas, M. Evaluation Metrics 2004. *INEX 2004 Workshop Preproceedings*, Schloss Dagstuhl, December 6-8, 2004.
- [9] Singhal, A; Buckley, C; Mitra, M. Pivoted Document Length Normalization. *ACM SIGIR*, 1996.

## Appendix A: Tables

### A.1 P@20 for Pseudo Feedback, at different values of M, num\_expand, $\alpha$ and $\beta$

Note: Values in bold indicate points at which P@20 for relevance feedback run > P@20 for basic run. Values in bold and underlined indicate the best value of precision for relevance feedback.

#### 1. M = 20

$(\alpha, \beta)$	5	10	25	50	100	300	500
<i>Basic Run</i>	0.3721	0.3721	0.3721	0.3721	0.3721	0.3721	0.3721
(1,1)	0.3324	0.3456	0.3588	<b>0.3838</b>	<b>0.3897</b>	<b>0.3838</b>	0.3632
(1,2)	0.3279	0.3441	0.3544	<b>0.3794</b>	<b>0.3750</b>	0.3691	0.3588
(1,3)	0.3279	0.3397	0.3544	<b>0.3750</b>	<b>0.3735</b>	0.3662	0.3544
(1,4)	0.3294	0.3412	0.3529	<b>0.3753</b>	<b>0.3721</b>	0.3662	0.3544
(1,5)	0.3294	0.3412	0.3515	<b>0.3735</b>	<b>0.3706</b>	0.3647	0.3544
(2,1)	0.3353	0.3559	0.3676	<b>0.3912</b>	<b>0.3941</b>	<b>0.3897</b>	<b>0.3765</b>
(2,3)	0.3309	0.3412	0.3544	<b>0.3794</b>	<b>0.3838</b>	<b>0.3750</b>	0.3588
(2,5)	0.3265	0.3397	0.3544	<b>0.3779</b>	<b>0.3750</b>	0.3676	0.3544
(3,1)	0.3397	0.3529	0.3480	<b>0.3956</b>	<b>0.4044</b>	<b>0.3941</b>	<b>0.3822</b>
(3,2)	0.3397	0.3485	0.3647	<b>0.3853</b>	<b>0.3868</b>	<b>0.3882</b>	<b>0.3721</b>
(3,4)	0.3309	0.3412	0.3544	<b>0.3809</b>	<b>0.3853</b>	<b>0.3765</b>	0.3588
(3,5)	0.3279	0.3412	0.3544	<b>0.3779</b>	<b>0.3794</b>	<b>0.3750</b>	0.3588
(4,1)	0.3426	0.3559	0.3691	<b>0.3956</b>	<u><b>0.4074</b></u>	<b>0.3942</b>	<b>0.4000</b>
(4,3)	0.3382	0.3471	0.3588	<b>0.3824</b>	<b>0.3897</b>	<b>0.3882</b>	0.3691
(4,5)	0.3324	0.3426	0.3544	<b>0.3809</b>	<b>0.3868</b>	<b>0.3765</b>	0.3574
(5,1)	0.3529	0.3544	0.3691	<b>0.4029</b>	<b>0.4074</b>	<b>0.4044</b>	<b>0.4029</b>
(5,2)	0.3368	0.3544	0.3691	<b>0.3956</b>	<b>0.3971</b>	<b>0.3912</b>	<b>0.3824</b>
(5,3)	0.3368	0.3471	0.3662	<b>0.3882</b>	<b>0.3897</b>	<b>0.3882</b>	<b>0.3735</b>
(5,4)	0.3368	0.3456	0.3574	<b>0.3824</b>	<b>0.3897</b>	<b>0.3853</b>	0.3676

Table A.1 P@20 for Pseudo Feedback and M=20

2. M = 10

$(\alpha, \beta)$	5	10	25	50	100	300	500
<i>Basic Run</i>	0.3721	0.3721	0.3721	0.3721	0.3721	0.3721	0.3721
(1,1)	0.3426	0.3456	0.3706	<b>0.3721</b>	<b>0.3735</b>	0.3632	0.3647
(1,2)	0.3265	0.3309	0.3632	<b>0.3721</b>	0.3618	0.3632	0.3574
(1,3)	0.3250	0.3253	0.3618	0.3500	0.3603	0.3471	0.3559
(1,4)	0.3250	0.3191	0.3603	0.3471	0.3588	0.3500	0.3529
(1,5)	0.3221	0.3147	0.3603	0.3471	0.3588	0.3485	0.3485
(2,1)	0.3426	0.3632	<b>0.3809</b>	<b>0.3824</b>	<b>0.3838</b>	0.3735	<b>0.3721</b>
(2,3)	0.3338	0.3338	0.3691	0.3618	0.3647	0.3588	0.3603
(2,5)	0.3250	0.3265	0.3647	0.3529	0.3603	0.3500	0.3559
(3,1)	0.3588	0.3632	<b>0.3809</b>	<b>0.3956</b>	<b>0.3956</b>	<b>0.3824</b>	<b>0.3882</b>
(3,2)	0.3412	0.3559	0.3706	<b>0.3765</b>	<b>0.3809</b>	0.3706	0.3662
(3,4)	0.3368	0.3397	0.3706	0.3662	0.3662	0.3588	0.3603
(3,5)	0.3309	0.3309	0.3676	0.3603	0.3647	0.3574	0.3588
(4,1)	0.3647	0.3618	<b>0.3853</b>	<b>0.3956</b>	<b>0.4059</b>	<b>0.4000</b>	<b>0.3926</b>
(4,3)	0.3441	0.3529	0.3706	<b>0.3809</b>	<b>0.3779</b>	0.3691	0.3632
(4,5)	0.3397	0.3397	0.3691	0.3662	0.3676	0.3603	0.3618
(5,1)	0.3691	0.3676	<b>0.3838</b>	<b>0.3956</b>	<b>0.4059</b>	<b>0.4059</b>	<b>0.4000</b>
(5,2)	0.3515	0.3603	<b>0.3824</b>	<b>0.3853</b>	<b>0.3912</b>	<b>0.3794</b>	<b>0.3868</b>
(5,3)	0.3441	0.3588	<b>0.3735</b>	<b>0.3794</b>	<b>0.3824</b>	<b>0.3750</b>	0.3706
(5,4)	0.3441	0.3529	0.3691	<b>0.3809</b>	<b>0.3750</b>	0.3647	0.3632

Table A.2 P@20 for Pseudo Feedback and M=10

### 3. M = 5

$(\alpha, \beta)$	5	10	25	50	100	300	500
<i>Basic Run</i>	0.3721	0.3721	0.3721	0.3721	0.3721	0.3721	0.3721
(1,1)	0.3618	0.3529	0.3632	0.3412	0.3544	0.3559	0.3426
(1,2)	0.3412	0.3426	0.3500	0.3250	0.3353	0.3382	0.3338
(1,3)	0.3382	0.3397	0.3500	0.3206	0.3309	0.3338	0.3309
(1,4)	0.3309	0.3382	0.3500	0.3147	0.3265	0.3294	0.3309
(1,5)	0.3235	0.3368	0.3441	0.3118	0.3265	0.3279	0.3294
(2,1)	<b>0.3779</b>	<b>0.3721</b>	<b>0.3824</b>	<b>0.3779</b>	<b>0.3794</b>	<b>0.3794</b>	<b>0.3779</b>
(2,3)	0.3515	0.3456	0.3544	0.3324	0.3441	0.3412	0.3441
(2,5)	0.3412	0.3412	0.3471	0.3235	0.3324	0.3353	0.3397
(3,1)	<b>0.3853</b>	<b>0.3779</b>	<b>0.3956</b>	<b>0.3882</b>	<b>0.3882</b>	<b>0.4015</b>	<b>0.4000</b>
(3,2)	<b>0.3735</b>	0.3662	<b>0.3779</b>	0.3603	0.3618	0.3691	0.3662
(3,4)	0.3559	0.3471	0.3529	0.3368	0.3515	0.3426	0.3485
(3,5)	0.3500	0.3485	0.3515	0.3324	0.3397	0.3382	0.3426
(4,1)	<b>0.3794</b>	<b>0.3853</b>	<b>0.3868</b>	<b>0.3912</b>	<b>0.3912</b>	<b>0.3912</b>	<b>0.3971</b>
(4,3)	0.3706	0.3603	<b>0.3765</b>	0.3544	0.3588	0.3618	0.3647
(4,5)	0.3574	0.3485	0.3559	0.3382	0.3515	0.3441	0.3515
(5,1)	<b>0.3794</b>	<b>0.3868</b>	<b>0.3824</b>	<b>0.3838</b>	<b>0.3926</b>	<b>0.3971</b>	<b>0.4029</b>
(5,2)	<b>0.3824</b>	<b>0.3794</b>	<b>0.3838</b>	<b>0.3838</b>	<b>0.3809</b>	<b>0.3882</b>	<b>0.3882</b>
(5,3)	<b>0.3735</b>	0.3676	<b>0.3750</b>	0.3676	0.3706	0.3706	0.3691
(5,4)	0.3706	0.3603	<b>0.3765</b>	0.3529	0.3588	0.3632	0.3618

Table A.3 P@20 for Pseudo Feedback and M=5

## A.2 P@20 for Relevance Feedback using Residual Collection, at different values of M, num\_expand, $\alpha$ and $\beta$

Note: Values in bold indicate points at which P@20 for relevance feedback run > P@20 for basic run. Values in bold and underlined indicate the best value of precision for relevance feedback.

### 1. M = 20

$(\alpha, \beta)$	5	10	25	50	100	300	500
<i>Basic Run</i>	0.3721	0.3721	0.3721	0.3721	0.3721	0.3721	0.3721
(1,1)	0.3676	<b>0.3941</b>	<b>0.3941</b>	<b>0.4059</b>	<b>0.4044</b>	<b>0.3912</b>	<b>0.3882</b>
(1,2)	0.3529	<b>0.3926</b>	<b>0.4044</b>	<b>0.3971</b>	<b>0.3971</b>	<b>0.3794</b>	<b>0.3809</b>
(1,3)	0.3544	<b>0.3882</b>	<b>0.4015</b>	<b>0.3971</b>	<b>0.3971</b>	<b>0.3794</b>	<b>0.3779</b>
(1,4)	0.3515	<b>0.3868</b>	<b>0.4044</b>	<b>0.4000</b>	<b>0.3941</b>	<b>0.3750</b>	<b>0.3750</b>
(1,5)	0.3500	<b>0.3868</b>	<b>0.4074</b>	<b>0.4000</b>	<b>0.3956</b>	<b>0.3750</b>	<b>0.3735</b>
(2,1)	0.3515	<b>0.3853</b>	<b>0.3971</b>	<b>0.4059</b>	<b>0.4029</b>	<b>0.3985</b>	<b>0.3971</b>
(2,3)	0.3529	<b>0.3941</b>	<b>0.4015</b>	<b>0.4044</b>	<b>0.4015</b>	<b>0.3838</b>	<b>0.3824</b>
(2,5)	0.3544	<b>0.3926</b>	<b>0.4000</b>	<b>0.3956</b>	<b>0.3956</b>	<b>0.3779</b>	<b>0.3779</b>
(3,1)	0.3529	<b>0.3765</b>	<b>0.3897</b>	<b>0.4059</b>	<b>0.4176</b>	<b>0.4044</b>	<b>0.4044</b>
(3,2)	0.3500	<b>0.3941</b>	<b>0.4000</b>	<b>0.4029</b>	<b>0.4000</b>	<b>0.3971</b>	<b>0.3882</b>
(3,4)	0.3559	<b>0.3956</b>	<b>0.4029</b>	<b>0.4044</b>	<b>0.4015</b>	<b>0.3882</b>	<b>0.3868</b>
(3,5)	0.3529	<b>0.3926</b>	<b>0.4015</b>	<b>0.3985</b>	<b>0.4029</b>	<b>0.3838</b>	<b>0.3824</b>
(4,1)	0.3515	<b>0.3676</b>	<b>0.3853</b>	<b>0.4147</b>	<b>0.4191</b>	<b>0.4000</b>	<b>0.4000</b>
(4,3)	0.3559	<b>0.3912</b>	<b>0.4015</b>	<b>0.4044</b>	<b>0.4015</b>	<b>0.3912</b>	<b>0.3882</b>
(4,5)	0.3574	<b>0.3956</b>	<b>0.4015</b>	<b>0.4000</b>	<b>0.4059</b>	<b>0.3912</b>	<b>0.3882</b>
(5,1)	0.3529	0.3647	0.3529	0.3647	<b>0.4103</b>	<b>0.4015</b>	<b>0.4029</b>
(5,2)	0.3529	<b>0.3824</b>	0.3529	<b>0.3824</b>	<b>0.4118</b>	<b>0.4044</b>	<b>0.4015</b>
(5,3)	0.3485	<b>0.3882</b>	0.3485	<b>0.3882</b>	<b>0.4015</b>	<b>0.4000</b>	<b>0.3962</b>
(5,4)	0.3603	<b>0.3912</b>	0.3603	<b>0.3912</b>	<b>0.4044</b>	<b>0.3897</b>	<b>0.3868</b>

Table A.4 P@20 for Relevance Feedback using Residual Collection and M=20

2. M = 10

$(\alpha, \beta)$	5	10	25	50	100	300	500
<i>Base Run</i>	0.3721	0.3721	0.3721	0.3721	0.3721	0.3721	0.3721
(1,1)	0.3676	<b>0.4074</b>	<b>0.4338</b>	<b>0.4324</b>	<b>0.4324</b>	<b>0.4221</b>	<b>0.4191</b>
(1,2)	0.3662	<b>0.4147</b>	<b>0.4265</b>	<b>0.4250</b>	<b>0.4265</b>	<b>0.4074</b>	<b>0.4059</b>
(1,3)	<b>0.3721</b>	<b>0.4088</b>	<b>0.4279</b>	<b>0.4221</b>	<b>0.4118</b>	<b>0.4044</b>	<b>0.4059</b>
(1,4)	0.3706	<b>0.4103</b>	<b>0.4235</b>	<b>0.4176</b>	<b>0.4132</b>	<b>0.3985</b>	<b>0.4000</b>
(1,5)	0.3691	<b>0.4059</b>	<b>0.4250</b>	<b>0.4147</b>	<b>0.4132</b>	<b>0.3971</b>	<b>0.3971</b>
(2,1)	<b>0.3735</b>	<b>0.3985</b>	<b>0.4250</b>	<b>0.4324</b>	<b>0.4324</b>	<b>0.4191</b>	<b>0.4191</b>
(2,3)	0.3674	<b>0.4176</b>	<b>0.4279</b>	<b>0.4353</b>	<b>0.4309</b>	<b>0.4103</b>	<b>0.4103</b>
(2,5)	0.3662	<b>0.4118</b>	<b>0.4250</b>	<b>0.4235</b>	<b>0.4147</b>	<b>0.4059</b>	<b>0.4059</b>
(3,1)	<b>0.3750</b>	<b>0.3882</b>	<b>0.4132</b>	<b>0.4265</b>	<b>0.4309</b>	<b>0.4265</b>	<b>0.4206</b>
(3,2)	<b>0.3735</b>	<b>0.4029</b>	<b>0.4279</b>	<b>0.4279</b>	<b>0.4309</b>	<b>0.4221</b>	<b>0.4191</b>
(3,4)	0.3618	<b>0.4147</b>	<b>0.4279</b>	<b>0.4353</b>	<b>0.4324</b>	<b>0.4147</b>	<b>0.4191</b>
(3,5)	0.3623	<b>0.4162</b>	<b>0.4279</b>	<b>0.4309</b>	<b>0.4294</b>	<b>0.4088</b>	<b>0.4074</b>
(4,1)	0.3691	<b>0.3809</b>	<b>0.4103</b>	<b>0.4132</b>	<b>0.4279</b>	<b>0.4132</b>	<b>0.4118</b>
(4,3)	<b>0.3706</b>	<b>0.4029</b>	<b>0.4353</b>	<b>0.4294</b>	<b>0.4338</b>	<b>0.4206</b>	<b>0.4206</b>
(4,5)	0.3623	<b>0.4088</b>	<b>0.4294</b>	<b>0.4353</b>	<b>0.4353</b>	<b>0.4162</b>	<b>0.4191</b>
(5,1)	0.3588	<b>0.3676</b>	<b>0.4088</b>	<b>0.4103</b>	<b>0.4235</b>	<b>0.4191</b>	<b>0.4147</b>
(5,2)	<b>0.3750</b>	<b>0.3912</b>	<b>0.4162</b>	<b>0.4309</b>	<b>0.4324</b>	<b>0.4265</b>	<b>0.4250</b>
(5,3)	<b>0.3750</b>	<b>0.3985</b>	<b>0.4265</b>	<b>0.4294</b>	<b>0.4309</b>	<b>0.4235</b>	<b>0.4191</b>
(5,4)	0.3676	<b>0.4074</b>	<b>0.4353</b>	<b>0.4309</b>	<b>0.4324</b>	<b>0.4206</b>	<b>0.4176</b>

Table A.5 P@20 for Relevance Feedback using Residual Collection and M=10

### 3. M = 5

$(\alpha, \beta)$	5	10	25	50	100	300	500
<i>Base Run</i>	0.3721	0.3721	0.3721	0.3721	0.3721	0.3721	0.3721
(1,1)	0.3853	0.4162	0.4235	0.4250	0.4324	0.4162	0.4162
(1,2)	0.3926	0.4235	0.4191	0.4353	0.4250	0.4074	0.4088
(1,3)	0.3926	0.4265	0.4235	0.4250	0.4191	0.3985	0.3985
(1,4)	0.3912	0.4235	0.4206	0.4206	0.4103	0.3941	0.3956
(1,5)	0.3868	0.4221	0.4206	0.4176	0.4059	0.3941	0.3956
(2,1)	0.3838	0.4000	0.4294	0.4368	0.4412	0.4338	0.4324
(2,3)	0.3926	0.4206	0.4191	0.4265	0.4324	0.4118	0.4118
(2,5)	0.3941	0.4265	0.4205	0.4294	0.4206	0.4015	0.4015
(3,1)	0.3853	0.3971	0.4103	0.4191	0.4338	0.4309	0.4324
(3,2)	0.3912	0.4103	0.4221	0.4338	0.4353	0.4221	0.4250
(3,4)	0.3882	0.4221	0.4191	0.4265	0.4382	0.4147	0.4147
(3,5)	0.3941	0.4206	0.4147	0.4294	0.4265	0.4103	0.4103
(4,1)	0.3721	0.3838	0.4044	0.4044	0.4118	0.4176	0.4191
(4,3)	0.3897	0.4147	0.4250	0.4309	0.4353	0.4279	0.4265
(4,5)	0.3912	0.4206	0.4191	0.4221	0.4324	0.4132	0.4118
(5,1)	0.3691	0.3868	0.4000	0.4000	0.4088	0.4118	0.4118
(5,2)	0.3853	0.3971	0.4206	0.4279	0.4338	0.4324	0.4324
(5,3)	0.3912	0.4044	0.4265	0.4338	0.4397	0.4221	0.4206
(5,4)	0.3882	0.4174	0.4279	0.4309	0.4353	0.4206	0.4191

Table A.6 P@20 for Relevance Feedback using Residual Collection and M=5

### A.3 P@20 for Relevance Feedback using Residual Collection with Overlap in Seen Elements Removed, at different values of num\_expand, $\alpha$ and $\beta$

Note: Values in bold indicate points at which P@20 for relevance feedback run > P@20 for basic run. Values in bold and underlined indicate the best value of precision for relevance feedback.

**M = 20**

<b>(<math>\alpha</math>, <math>\beta</math>)</b>	<b>5</b>	<b>10</b>	<b>25</b>	<b>50</b>	<b>100</b>	<b>300</b>	<b>500</b>
<b>Base Run</b>	<i>0.3721</i>	<i>0.3721</i>	<i>0.3721</i>	<i>0.3721</i>	<i>0.3721</i>	<i>0.3721</i>	<i>0.3721</i>
<b>(1,1)</b>	0.2779	0.3264	0.3441	0.3691	0.3632	0.3676	0.3676
<b>(1,2)</b>	0.2647	0.3205	0.3411	0.3558	0.3573	0.3558	0.3573
<b>(1,3)</b>	0.2602	0.3161	0.3367	0.3602	0.3514	0.3514	0.3529
<b>(1,4)</b>	0.2617	0.3058	0.3367	0.3544	0.3470	0.3441	0.3470
<b>(1,5)</b>	0.2588	0.2955	0.3352	0.3544	0.3485	0.3411	0.3441
<b>(2,1)</b>	0.2779	0.3161	0.3382	0.3632	0.3705	<b>0.3779</b>	<b>0.3764</b>
<b>(2,3)</b>	0.2735	0.3294	0.3411	0.3647	0.3544	0.3588	0.3573
<b>(2,5)</b>	0.2647	0.3205	0.3352	0.3573	0.3558	0.3558	0.3573
<b>(3,1)</b>	0.2735	0.3117	0.3367	0.3514	<b>0.3764</b>	<b>0.3764</b>	<b><u>0.3852</u></b>
<b>(3,2)</b>	0.2779	0.3205	0.3441	0.3735	0.3647	<b>0.3779</b>	0.3691
<b>(3,4)</b>	0.2779	0.3308	0.3426	0.3691	0.3573	0.3602	0.3602
<b>(3,5)</b>	0.2705	0.3279	0.3441	0.3602	0.3573	0.3602	0.3602
<b>(4,1)</b>	0.2676	0.3029	0.3382	0.3529	0.3691	<b>0.3779</b>	<b>0.3838</b>
<b>(4,3)</b>	0.2794	0.3220	0.3411	0.3691	0.3617	0.3676	0.3661
<b>(4,5)</b>	0.2779	0.3294	0.3441	0.3691	0.3558	0.3602	0.3617
<b>(5,1)</b>	0.2661	0.3000	0.3250	0.3558	0.3676	<b>0.3735</b>	<b>0.3779</b>
<b>(5,2)</b>	0.2750	0.3147	0.3397	0.3602	<b>0.3779</b>	<b>0.3808</b>	<b>0.3882</b>
<b>(5,3)</b>	0.2779	0.3161	0.3411	0.3676	0.3647	<b>0.3808</b>	<b>0.3720</b>
<b>(5,4)</b>	0.2779	0.3220	0.3411	0.3691	0.3617	0.3676	0.3661

Table A.7 P@20 for Relevance Feedback using Residual Collection and with Overlap in Seen Elements Removed, for M=20

## A.4 Overlap in Retrieval Result

The tables below show the number of elements of each type in a retrieval result, averaged over all experimental queries.  $D$  is the number of distinct documents represented in the retrieval result,  $P$  the number of paragraphs,  $SS$  the number of sub-sections,  $S$  the number of sections,  $B$  the number of bodies (whole document) and  $O$  the number of elements other than the ones mentioned here.

### 1. Initial (Basic) Retrieval

<b>D</b>	<b>P</b>	<b>SS</b>	<b>S</b>	<b>B</b>	<b>O</b>
12	7	4	8	0	1

Table A.8 Number of elements of each type in the basic retrieval

**2. Relevance Feedback using Residual Collection with and without Overlap (and descendents of) Seen Elements Removed.**

**a. num\_expand = 5, M = 20**

<b>(<math>\alpha</math>, <math>\beta</math>)</b>	<b>Overlap Not Removed</b>	<b>Overlap Removed</b>
	D P S S S B O	D P S S S B O
<b>(1,1)</b>	14 4 3 11 1 1	15 3 3 12 1 1
<b>(1,2)</b>	14 4 3 11 1 1	15 3 3 12 1 1
<b>(1,3)</b>	14 4 3 11 1 1	15 3 3 12 1 1
<b>(1,4)</b>	14 4 3 11 1 1	14 3 3 12 1 1
<b>(1,5)</b>	14 4 3 11 1 1	14 3 3 12 1 1
<b>(2,1)</b>	14 5 3 11 1 1	15 3 3 12 1 1
<b>(2,3)</b>	14 4 3 11 1 1	15 3 3 12 1 1
<b>(2,5)</b>	14 4 3 11 1 1	15 3 3 12 1 1
<b>(3,1)</b>	14 5 3 10 1 1	15 3 3 12 1 1
<b>(3,2)</b>	14 5 3 11 1 1	15 3 3 12 1 1
<b>(3,4)</b>	14 4 3 11 1 1	15 3 3 12 1 1
<b>(3,5)</b>	14 4 3 11 1 1	15 3 3 12 1 1
<b>(4,1)</b>	14 5 3 10 1 1	15 3 3 12 1 1
<b>(4,3)</b>	14 5 3 11 1 1	15 3 3 12 1 1
<b>(4,5)</b>	14 4 3 11 1 1	15 3 3 12 1 1
<b>(5,1)</b>	14 5 3 10 1 1	15 3 3 11 1 1
<b>(5,2)</b>	14 5 3 10 1 1	15 3 3 11 1 1
<b>(5,3)</b>	14 5 3 11 1 1	15 3 3 12 1 1
<b>(5,4)</b>	14 5 3 10 1 1	15 3 3 12 1 1

Table A.9 Number of elements of each type, for num\_expand = 5

b. num\_expand = 10, M = 20

$(\alpha, \beta)$	Overlap Not Removed	Overlap Removed
	D P S S S B O	D P S S S B O
(1,1)	14 3 3 11 2 1	15 2 3 13 2 1
(1,2)	14 3 3 11 2 1	14 2 3 13 2 1
(1,3)	14 3 3 11 2 1	14 2 3 13 2 1
(1,4)	14 3 3 11 2 1	14 2 3 13 2 1
(1,5)	14 3 3 11 2 1	14 2 3 13 2 1
(2,1)	14 3 3 11 1 1	15 2 3 13 2 1
(2,3)	14 3 3 11 2 1	14 2 3 13 2 1
(2,5)	14 3 3 12 2 1	14 2 3 13 2 1
(3,1)	14 4 3 11 1 1	15 2 3 13 1 1
(3,2)	14 3 3 11 2 1	15 2 3 13 2 1
(3,4)	14 3 3 11 2 1	14 2 3 13 2 1
(3,5)	14 3 3 11 2 1	14 2 3 13 2 1
(4,1)	14 4 3 10 1 1	15 2 3 12 1 1
(4,3)	14 3 3 11 2 1	15 2 3 13 2 1
(4,5)	14 3 3 11 2 1	14 2 3 13 2 1
(5,1)	14 5 3 10 1 1	15 2 3 12 1 1
(5,2)	14 4 3 11 1 1	15 2 3 13 2 1
(5,3)	14 3 3 11 2 1	15 2 3 13 2 1
(5,4)	14 3 3 11 2 1	15 2 3 13 2 1

Table A.10 Number of elements of each type, for num\_expand = 10

c. num\_expand = 25, M = 20

$(\alpha, \beta)$	Overlap Not Removed	Overlap Removed
	D P S S S B O	D P S S S B O
(1,1)	14 2 3 12 3 1	14 1 2 13 3 0
(1,2)	13 2 3 11 3 1	14 1 2 13 3 0
(1,3)	13 2 3 11 3 1	14 1 2 13 3 0
(1,4)	13 2 3 11 3 0	14 1 2 13 3 0
(1,5)	13 2 3 11 3 0	15 1 2 13 4 0
(2,1)	14 3 3 11 2 1	15 1 3 13 3 1
(2,3)	13 2 3 12 3 1	14 1 2 13 3 0
(2,5)	13 2 3 11 3 1	14 1 2 13 3 0
(3,1)	14 3 3 11 2 1	15 1 3 13 3 1
(3,2)	14 2 3 12 3 1	15 1 3 13 3 1
(3,4)	13 2 3 12 3 1	14 1 2 13 3 0
(3,5)	13 2 3 12 3 1	14 1 2 13 3 0
(4,1)	14 3 3 11 2 1	15 1 3 13 2 1
(4,3)	14 2 3 12 3 1	15 1 2 13 3 0
(4,5)	14 2 3 12 3 1	14 1 2 13 3 0
(5,1)	14 3 3 11 2 1	15 2 3 13 2 1
(5,2)	14 3 3 11 2 1	15 1 3 12 3 1
(5,3)	14 3 3 12 3 1	15 1 3 13 3 1
(5,4)	14 2 3 12 3 1	14 1 2 13 3 1

Table A.11 Number of elements of each type, for num\_expand = 25

d. num\_expand = 50, M = 20

$(\alpha, \beta)$	Overlap Not Removed	Overlap Removed
	D P S S S B O	D P S S S B O
(1,1)	14 2 2 11 5 0	14 1 2 11 6 0
(1,2)	14 2 2 11 6 0	14 1 1 11 6 0
(1,3)	14 2 2 11 6 0	14 1 1 11 7 0
(1,4)	14 2 2 10 6 0	14 1 1 11 7 0
(1,5)	14 2 2 10 6 0	14 1 1 11 7 0
(2,1)	14 2 2 10 5 0	15 1 2 12 5 0
(2,3)	14 2 2 11 5 0	14 1 1 11 6 0
(2,5)	14 2 2 11 6 0	14 1 1 11 6 0
(3,1)	14 2 3 11 4 0	15 1 2 12 5 0
(3,2)	14 2 2 11 5 0	15 1 2 11 6 0
(3,4)	14 2 2 11 5 0	15 1 1 11 6 0
(3,5)	14 2 2 11 5 0	14 1 1 11 6 0
(4,1)	14 2 3 11 4 0	15 1 2 12 5 0
(4,3)	14 2 2 11 5 0	15 1 2 12 6 0
(4,5)	14 2 2 11 5 0	14 1 2 11 6 0
(5,1)	14 2 3 11 4 0	15 1 2 12 4 0
(5,2)	14 2 3 11 5 0	15 1 2 12 5 0
(5,3)	14 2 2 11 5 0	15 1 2 12 6 0
(5,4)	14 2 2 11 5 0	15 1 2 12 6 0

Table A.12 Number of elements of each type, for num\_expand = 50

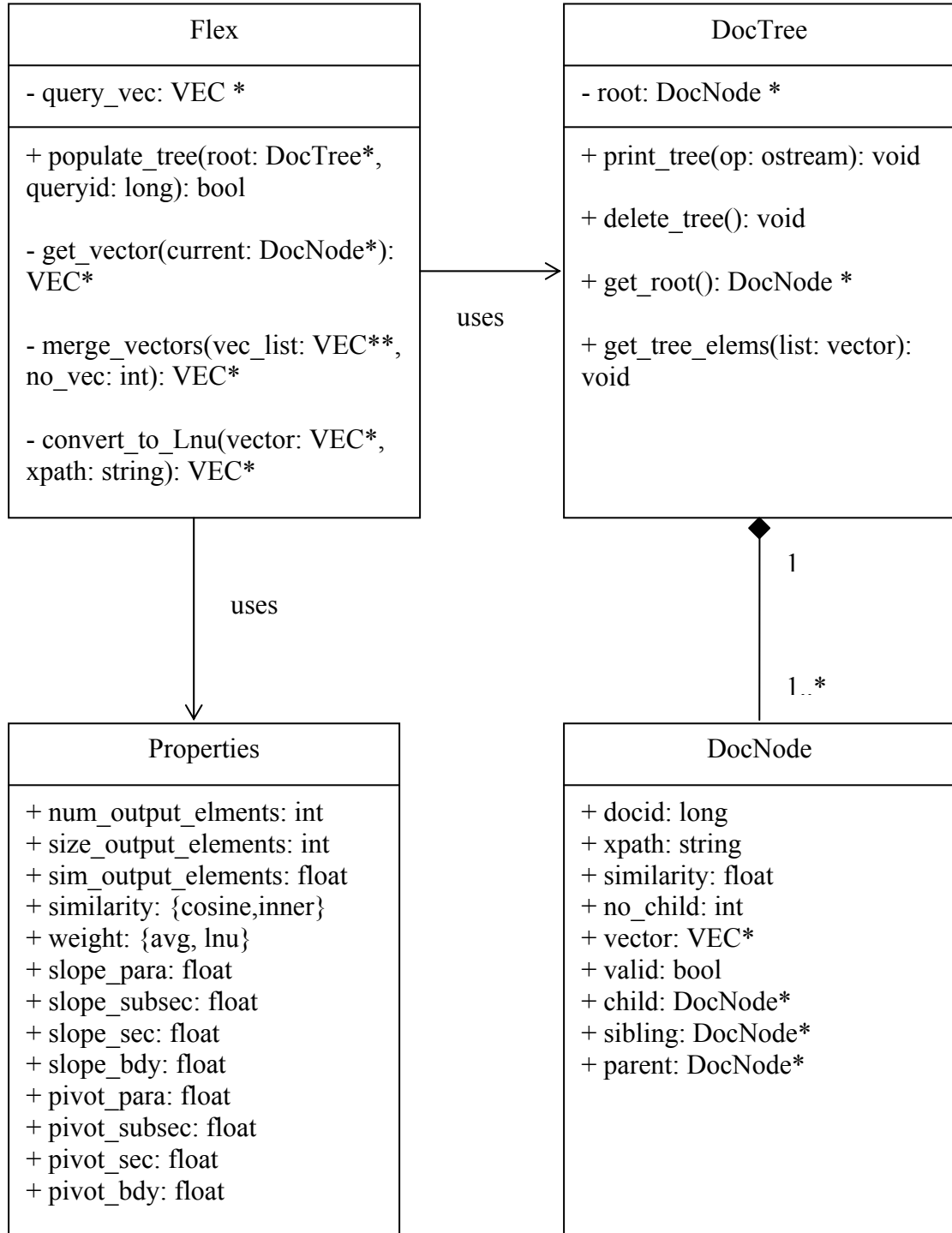
e. num\_expand = 500, M = 20

$(\alpha, \beta)$	Overlap Not Removed	Overlap Removed
	D P S S S B O	D P S S S B O
(1,1)	14 2 1 9 8 0	15 1 1 9 9 0
(1,2)	14 2 1 9 8 0	15 1 1 9 9 0
(1,3)	14 2 1 9 8 0	15 1 1 9 9 0
(1,4)	14 2 1 9 8 0	15 1 1 9 9 0
(1,5)	14 2 1 9 8 0	15 1 1 9 9 0
(2,1)	14 2 2 9 7 0	15 1 1 10 8 0
(2,3)	14 2 1 9 8 0	15 1 1 9 9 0
(2,5)	14 2 1 9 8 0	15 1 1 9 9 0
(3,1)	14 2 2 10 6 0	15 1 2 10 7 0
(3,2)	14 2 2 9 7 0	15 1 1 10 8 0
(3,4)	14 2 1 9 8 0	15 1 1 9 9 0
(3,5)	14 2 1 9 8 0	15 1 1 9 9 0
(4,1)	14 2 2 10 6 0	15 1 2 10 7 0
(4,3)	14 2 2 9 7 0	15 1 1 9 8 0
(4,5)	14 2 1 9 8 0	15 1 1 9 9 0
(5,1)	14 2 2 10 5 0	15 1 2 11 6 0
(5,2)	14 2 2 10 6 0	15 1 1 10 8 0
(5,3)	14 2 2 9 7 0	15 1 1 10 8 0
(5,4)	14 2 1 9 7 0	15 1 1 9 9 0

Table A.13 Number of elements of each type, for num\_expand = 500

## Appendix B: Flex Class and Sequence Diagram

### Class Diagram



### Sequence Diagram

