

UNIVERSITY OF MINNESOTA

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

Harsh Bapat

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

Adapting the Extended Vector Space Model for Structured XML Retrieval

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

Harsh Bapat

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

August 2003

Department of Computer Science
University of Minnesota Duluth
Duluth, Minnesota 55812
U.S.A.

© Harsh Bapat 2003

Abstract

Information retrieval focuses on retrieving *relevant* information from the available information. With the widespread use of XML in digital libraries, scientific data repositories, and XML emerging as future document standard for the Web, it is of natural interest to the information retrieval community. XML is highly structured language used to represent the logical structure of a document. One can represent different classes of information such as article title, author name, abstract, bibliography, etc. in a XML document. These different classes of information within the document are distinctly identifiable by the XML elements. The highly structured nature of XML allows more specific and complex search strategies in the form of content-and-structure (CAS) and content-only (CO) queries. CAS queries can restrict the context of interest or context of a particular search words to a part of the document. CO queries are like traditional queries without structural constraints on search words, but expect the retrieval of most relevant document component rather than the document itself. Thus we have at hand a well-defined retrieval task in the form of XML documents and CAS and CO queries which is significantly different and complex than the traditional retrieval task.

In our thesis we propose the use of extended vector space model, with some modifications, for this XML retrieval task. Experimental results compare favorably to those of other researchers and indicate that the extended vector space model provides a natural framework for structured XML retrieval.

Acknowledgements

I owe sincere thanks to my advisor Dr. Carolyn Crouch for her guidance throughout my graduate career, for imparting valuable knowledge in the Information Retrieval field, for teaching me the ABCs of research, for encouraging me time and again, and for all her words of kindness.

I am also very thankful to Dr. Donald Crouch and Dr. Douglas Dunham who gave me invaluable feedback on my thesis. I specially acknowledge Dr. Dunham for agreeing to serve on master's thesis committee on a short notice.

Thanks due to Steve Holtz who answered my unending questions regarding the Smart system. I would like to especially thank Sameer Apte for his cooperation when we worked as a team. I would like to thank Aniruddha Mahajan and Archana Bellmkonda for their help with programs.

I would like to acknowledge the help of Department of Computer Science at University of Minnesota Duluth. Specifically I would like to thank Lori Lucia, Linda Meek and Jim Luttinen for help with infrastructure.

I would like to thank my family members and friends for keeping my spirits high through this arduous journey. Finally, my thanks to my parents without whose support and inspiration I could not have accomplished this Master's degree. And most of all, I thank my wife Swati Bapat, for inspiring and encouraging me and putting up with me when I was not by her side.

Table of Contents

1	Introduction	vi
1.1	Related Work.....	2
1.1.1	IR Model-oriented Approach	2
1.1.2	Database-oriented Approach	2
1.1.3	XML-specific Approach.....	2
1.2	Overview of thesis.....	3
2	Data and Evaluation Procedures for Experimentation	4
2.1	Data for Experimentation	4
2.1.1	Documents (articles).....	4
2.1.2	Queries (topics)	6
2.2	Assessments.....	9
2.2.1	Implicit Assessments	10
2.3	Evaluation Metrics.....	11
2.3.1	Quantisation of Relevance and Coverage.....	11
2.3.2	Recall/Precision Metrics.....	12
3	Adapting the Extended Vector Space Model to CAS Queries	16
3.1	Vector Space Model	16
3.2	Extended Vector Space Model	17
3.3	Adapting the Extended Vector Space Model to CAS Queries	18
3.4	Implementation Details	21
3.4.1	The Pre-parser	22
3.4.2	The Result Merger (Query Resolution).....	25
3.4.3	Result Conversion	25
4	Experiments.....	29
4.1	The Setup.....	29
4.2	Initial Experiments	29
4.3	Experiments with Different Term Weighting Schemes	30
4.3.1	<i>Lnu-ltu</i> Weighting Scheme.....	30

4.3.2	<i>atc</i> Weighting Scheme.....	32
4.4	Weighting amongst the Subvectors	35
5	Conclusions and Future Work	38
5.1	Conclusions	38
5.2	Future Work.....	39
6	Bibliography	41
7	Appendix	44
7.1	Recall/Precision curves	44
7.1.1	Recall/Precision curves for our Initial Experiment	44
7.1.2	Recall/Precision curves for Experiments with Different Weighting Schemes.....	45
7.1.3	Recall/Precision curves for Weighting amongst the Subvectors.....	50

List of Tables

Table 1: The INEX document collection statistics.....	5
Table 2: C-types and their description.....	19
Table 3: Expat parser handlers and their functions	23
Table 4: Comparison of our results for variation of <i>Lnu-ltu</i> weighting scheme to those reported at INEX 2002 for generalized quantisation	31
Table 5: Comparison of our results for variation of <i>Lnu-ltu</i> weighting scheme to those reported at INEX 2002 for strict quantisation.....	31
Table 6: Comparison of our results for variation of <i>atc</i> weighting scheme to those reported at INEX 2002 for generalized quantisation	33
Table 7: Comparison of our results for variation of <i>atc</i> weighting scheme to those reported at INEX 2002 for strict quantisation.....	33
Table 8: Comparison of results for different weighting schemes under generalized quantisation	34
Table 9: Comparison of results for different weighting schemes under strict quantisation	34
Table 10: Effect of changing weights of abstract, article title, and keywords subvectors by keeping the weight of body subvector constant at 1.0.....	36
Table 11: Effect of changing weight of abstract, article title, and keywords subvectors one by one to lower value and keeping the weight of body subvector constant at 4.0	37
Table 12: Effect of changing weight of abstract, article title, and keywords subvectors one by one to higher value and keeping the weight of body subvector constant at 4.0	37
Table 13: Results with higher weights for body subvector and lower weights for abstract, article title, and keywords subvectors	37

List of Figures

Figure 1: Sketch of the structure of the typical INEX articles	6
Figure 2: A CAS topic from the INEX test collection	7
Figure 3: An Example of Vector Space Model	17
Figure 4: An Example of an Extended Vector	18
Figure 5: Stack contents at different stages of parsing.....	24
Figure 6: Result merger performs AND operation on the split queries lists to produce a final list. Final ranking follows the ranking of the subjective part of the query.	26
Figure 7: INEX retrieval result submission format DTD.....	27
Figure 8: Example INEX retrieval result submission.....	27
Figure 9: Recall/Precision curve for generalized quantisation – Initial results with <i>Lnu- ltu</i> weights to all subvectors.....	44
Figure 10: Recall/Precision curves for generalized quantization – <i>Lnu-ltu</i> weights to all subvectors.....	45
Figure 11: Recall/Precision curves for strict quantisation – <i>Lnu-ltu</i> weights to all subvectors.....	45
Figure 12: Recall/Precision curves for generalized quantisation – <i>nnn</i> weights to objective subvectors and <i>Lnu-ltu</i> weights to subjective subvectors.....	46
Figure 13: Recall/Precision curves for strict quantisation – <i>nnn</i> weights to objective subvectors and <i>Lnu-ltu</i> weights to subjective subvectors.....	46
Figure 14: Recall/Precision curve for generalized quantisation – <i>atc</i> weights to all subvectors.....	47
Figure 15: Recall/Precision curve for strict quantisation – <i>atc</i> weights to all subvectors	47
Figure 16: Recall/Precision curve for generalized quantisation – <i>nnn</i> weights to objective subvectors and <i>atc</i> weights to subjective subvectors.....	48
Figure 17: Recall/Precision curve for strict quantisation – <i>nnn</i> weights to objective subvectors and <i>atc</i> weights to subjective subvectors	48

Figure 18: Recall/Precision curve for generalized quantisation – <i>nnn</i> weights to objective subvectors, <i>Lnu-ltu</i> weights to body subvector and <i>atc</i> weights to rest of the subvectors	49
Figure 19: Recall/Precision curve for strict quantisation – <i>nnn</i> weights to objective subvectors, <i>Lnu-ltu</i> weights to body subvector and <i>atc</i> weights to rest of the subvectors	49
Figure 20: Recall/Precision curve for generalized quantisation with <i>atc</i> weights to all subvectors– weighting amongst the subvectors: <i>abstract=0.5, article title=0.5, keywords=0.5, body=1.0</i>	50
Figure 21: Recall/Precision curve for strict quantisation with <i>atc</i> weights to all subvectors– weighting amongst the subvectors: <i>abstract=0.5, article title=0.5, keywords=0.5, body=1.0</i>	50
Figure 22: Recall/Precision curve for generalized quantisation with <i>atc</i> weights to all subvectors– weighting amongst the subvectors: <i>abstract=2.0, article title=2.0, keywords=2.0, body=1.0</i>	51
Figure 23: Recall/Precision curve for strict quantisation with <i>atc</i> weights to all subvectors– weighting amongst the subvectors: <i>abstract=2.0, article title=2.0, keywords=2.0, body=1.0</i>	51
Figure 24: Recall/Precision curve for generalized quantisation with <i>atc</i> weights to all subvectors– weighting amongst the subvectors: <i>abstract=3.0, article title=3.0, keywords=3.0, body=1.0</i>	52
Figure 25: Recall/Precision curve for strict quantisation with <i>atc</i> weights to all subvectors– weighting amongst the subvectors: <i>abstract=3.0, article title=3.0, keywords=3.0, body=1.0</i>	52
Figure 26: Recall/Precision curve for generalized quantisation with <i>atc</i> weights to all subvectors– weighting amongst the subvectors: <i>abstract=0.2, article title=1.0, keywords=1.0, body=4.0</i>	53
Figure 27: Recall/Precision curve for strict quantisation with <i>atc</i> weights to all subvectors– weighting amongst the subvectors: <i>abstract=0.2, article title=1.0, keywords=1.0, body=4.0</i>	53

Figure 28: Recall/Precision curve for generalized quantisation with <i>atc</i> weights to all subvectors– weighting amongst the subvectors: <i>abstract=1.0, article title=0.2, keywords=1.0, body=4.0</i>	54
Figure 29: Recall/Precision curve for strict quantisation with <i>atc</i> weights to all subvectors– weighting amongst the subvectors: <i>abstract=1.0, article title=0.2, keywords=1.0, body=4.0</i>	54
Figure 30: Recall/Precision curve for generalized quantisation with <i>atc</i> weights to all subvectors– weighting amongst the subvectors: <i>abstract=1.0, article title=1.0, keywords=0.2, body=4.0</i>	55
Figure 31: Recall/Precision curve for strict quantisation with <i>atc</i> weights to all subvectors– weighting amongst the subvectors: <i>abstract=1.0, article title=1.0, keywords=0.2, body=4.0</i>	55
Figure 32: Recall/Precision curve for generalized quantisation with <i>atc</i> weights to all subvectors– weighting amongst the subvectors: <i>abstract=1.0, article title=0.2, keywords=0.2, body=4.0</i>	56
Figure 33: Recall/Precision curve for strict quantisation with <i>atc</i> weights to all subvectors– weighting amongst the subvectors: <i>abstract=1.0, article title=0.2, keywords=0.2, body=4.0</i>	56
Figure 34: Recall/Precision curve for generalized quantisation with <i>atc</i> weights to all subvectors– weighting amongst the subvectors: <i>abstract=0.2, article title=1.0, keywords=0.2, body=4.0</i>	57
Figure 35: Recall/Precision curve for strict quantisation with <i>atc</i> weights to all subvectors– weighting amongst the subvectors: <i>abstract=0.2, article title=1.0, keywords=0.2, body=4.0</i>	57
Figure 36: Recall/Precision curve for generalized quantisation with <i>atc</i> weights to all subvectors– weighting amongst the subvectors: <i>abstract=0.2, article title=0.2, keywords=1.0, body=4.0</i>	58
Figure 37: Recall/Precision curve for strict quantisation with <i>atc</i> weights to all subvectors– weighting amongst the subvectors: <i>abstract=0.2, article title=0.2, keywords=1.0, body=4.0</i>	58

Figure 38: Recall/Precision curve for generalized quantisation with <i>atc</i> weights to all subvectors– weighting amongst the subvectors: <i>abstract=0.2, article title=0.2, keywords=0.2, body=4.0</i>	59
Figure 39: Recall/Precision curve for strict quantisation with <i>atc</i> weights to all subvectors– weighting amongst the subvectors: <i>abstract=0.2, article title=0.2, keywords=0.2, body=4.0</i>	59
Figure 40: Recall/Precision curve for generalized quantisation with <i>atc</i> weights to all subvectors– weighting amongst the subvectors: <i>abstract=0.01, article title=0.01, keywords=0.01, body=8.0</i>	60
Figure 41: Recall/Precision curve for strict quantisation with <i>atc</i> weights to all subvectors– weighting amongst the subvectors: <i>abstract=0.01, article title=0.01, keywords=0.01, body=8.0</i>	60
Figure 42: Recall/Precision curve for generalized quantisation with <i>atc</i> weights to all subvectors– weighting amongst the subvectors: <i>abstract=1.0, article title=1.0, keywords=1.0, body=8.0</i>	61
Figure 43: Recall/Precision curve for strict quantisation with <i>atc</i> weights to all subvectors– weighting amongst the subvectors: <i>abstract=1.0, article title=1.0, keywords=1.0, body=8.0</i>	61

1 Introduction

Information retrieval focuses on retrieving *relevant* information from the available information. With the exponential growth of the *information* available on the web and the XML becoming the future standard for the representation of web documents, it is of natural interest to the information retrieval community.

XML is a highly structured language and used to represent the logical structure of a document. XML uses opening and closing tags, also called as elements, to mark the boundary of a logical part of a document. For example, when representing a research paper in XML, an abstract of the research paper may be enclosed within the opening tag `<abstract>` and the end tag `</abstract>`, the name of the author of the paper can be enclosed within tags `<author>` and `</author>`, etc.

This structural nature of XML gives the user of a XML retrieval system the ability to issue more complex and precise queries than those used in traditional flat (unstructured) document retrieval. Users can make use of the structural nature of XML documents to restrict their search to specific structural elements within the XML document collection. For example, it is possible to search within a collection of research papers in XML format, all papers written by a specific author by restricting our search to the `<author>` element. Such queries are termed as *content-and-structure* (CAS) queries by the XML retrieval community [9].

Though CAS queries are more complex than traditional queries, they are potentially more powerful. The result of retrieval for a traditional query is a complete document. In the case of CAS queries (structured retrieval) the result can be any part of the document logically identified by the XML elements. The user can restrict the context of interest or the context of certain search words by explicitly confining the search words to a structural part of the XML document. For example, a CAS query

`<author>Gerard Salton</author>`

<abstract>vector space model</abstract>

is expected to fetch those documents whose abstract refers the vector space model which are authored by Gerard Salton. One drawback of such queries, as opposed to the traditional queries, is that user must know the document structure beforehand.

1.1 Related Work

There are number of heterogeneous research approaches to solving this problem of structured retrieval. They can be grouped into three major categories [9]:

1. IR model-oriented,
2. Database-oriented, and
3. XML-specific.

1.1.1 IR Model-oriented Approach

The field of information retrieval has grown significantly over the past 20 to 30 years. Numerous IR models such as the vector-space, probability and rule-based models and systems implementing those models are currently in place. Conventional information retrieval models are designed for text retrieval where the documents and queries are plain text (or what we call unstructured). These systems cannot be directly used for the XML retrieval task. In this approach we use an extension of a specific information retrieval model to deal with structured XML documents and queries.

1.1.2 Database-oriented Approach

Database management systems are the most popular retrieval systems for XML documents. But the nature of XML documents is not content oriented. To deal with content-oriented XML documents, database management systems have undergone modifications as described in [1].

1.1.3 XML-specific Approach

IR model-oriented and database approaches extend traditional retrieval models to adapt them to the XML retrieval task. The XML specific approach is based on the models and systems developed specifically for the XML retrieval task. Most of the models are based on existing XML standards such as XPath [20] and XQuery [21]. XPath is a

language for addressing the parts of an XML document while XQuery is a query language used to query all types of XML data sources. One other technique not based on the XML standards is Extreme file inversion (EFI) [8]. EFI uses the file inversion technique, where the location of the words within the document is also stored for context searching.

Some researchers also use a combination of the above-described approaches.

1.2 Overview of thesis

We address the structured retrieval problem by proposing the use of extended vector space model [7] for the structured retrieval task and show that the extended vector space model provides a natural framework for structured retrieval.

We evaluate our approach by comparing our results to those attained at the INEX 2002 workshop. INEX, the Initiative for the Evaluation of XML Retrieval [14], is an international effort supporting research in information retrieval and digital libraries. It promotes the evaluation of content-based XML retrieval by making available an XML test collection and evaluation procedures. We participated in the building of the test collection for INEX 2002 and used it for our experiments.

This thesis continues with a detailed description of the INEX 2002 test collection and evaluation procedures. This is followed by a description of the extended vector space model and its adaptation to the CAS queries task. Next we describe our experiments and their evaluations. Finally we present our conclusions and discuss possible future work.

2 Data and Evaluation Procedures for Experimentation

For our experiments we used the test collection developed as a part of the INEX 2002 [14] workshop, which promotes the development and evaluation of content-based XML retrieval systems. To evaluate our approach we follow the evaluation procedures developed at INEX 2002 and implemented in the `inex_eval` package. The test collection and the `inex_eval` package were distributed freely to the participants of the INEX 2002 workshop. We were participants in the workshop and also contributed to the development and assessments of CAS topics. In the following sections we describe the test collection, which consists of XML documents and the topics, and the evaluation procedures.

2.1 Data for Experimentation

The INEX 2002 test collection is similar to a standard IR test collection. It consists of three parts, namely a set of documents, topics or queries, and relevance assessments.

2.1.1 Documents (articles)

The document collection is made up of full texts of articles from 12 magazines and 6 transactions of the IEEE Computer Society's publications. In all, there are 12,107 articles from 1995-2002, with a size of 494 MB. Table 1 gives some statistics of the document collection. Although the document collection is small in size as compared to the TREC [19] collections, it has a complex XML structure containing 192 different content models in its DTD. On average, an article contains 1,532 XML nodes, where the average depth of a node is 6.9.

All the articles are marked up in XML and follow a common schema, i.e., DTD. Figure 1 shows the overall structure of a typical article. A typical article consists of a front matter (`<fm>`), a body (`<bdy>`), and a back matter (`<bm>`). The front matter contains the article's metadata, such as title, author, publication information and abstract. The body contains the main text of the article. It is structured into sections (`<sec>`), sub-

sections (<ss1>), and sub-sub-sections (<ss2>). These structures start with a title followed by paragraphs (<p>) within them. The back matter contains a bibliography and information about the authors of the articles.

Table 1: The INEX document collection statistics

Id	Publication title	Year	Size	No of articles
An	IEEE Annals of the History of Computing	1995-2001	13.2	316
Cg	IEEE Computer Graphics and Applications	1995-2001	19.1	680
Co	Computer	1995-2001	40.4	1902
Cs	IEEE Computational Science and Engineering	1995-1998	14.6	571
	Computing in Science and Engineering	1999-2001		
Dt	IEEE Design & Test of Computers	1995-2001	13.6	539
Ex	IEEE Expert	1995-1997	20.3	702
	IEEE Intelligent Systems	1998-2001		
Ic	IEEE Internet Computing	1997-2001	12.2	547
It	IT Professional	1999-2001	4.7	249
Mi	IEEE Micro	1995-2001	15.8	604
Mu	IEEE Multimedia	1995-2001	11.3	465
Pd	IEEE Parallel & Distributed Technology	1996-1996	10.7	363
	IEEE Concurrency	1997-2000		
So	IEEE Software	1995-2001	20.9	936
Tc	IEEE Transactions on Computers	1995-2002	66.1	1042
Td	IEEE Transactions on Parallel & Distributed Systems	1995-2002	58.8	765
Tg	IEEE Transactions on Visualization & Computer Graphics	1995-2002	15.2	225
Tk	IEEE Transactions on Knowledge and Data Engineering	1995-2002	48.1	585
Tp	IEEE Transactions on Pattern Analysis & Machine Intelligence	1995-2002	62.9	1046
Ts	IEEE Transactions on Software Engineering	1995-2002	46.1	570
Total			494	12, 107

```

<article>
  <fm>
    ...
    <ti>IEEE Transactions on ...</ti>
    <atl>Construction of ...</atl>
    <au>
      <fnm>John</fnm>
      <snm>Smith</snm>
      <aff>University of ...</aff>
    </au>
    ...
  </fm>
  <bdy>
    <sec>
      <st>Introduction</st>
      <p>...</p>
      ...
    </sec>
    <sec>
      <st>...</st>
      ...
      <ss1>...</ss1>
      ...
    </sec>
    ...
  </bdy>
  <bm>
    <bib>
      <bb>
        <au>...</au><ti>...</ti>
        ...
      </bb>
      ...
    </bib>
  </bm>
</article>

```

Figure 1: Sketch of the structure of the typical INEX articles

2.1.2 Queries (topics)

The test collection contains two types of topics or queries: *content-and-structure* (CAS) queries and *content-only* (CO) queries. The workshop participants developed these topics. The topic format and the topics development procedures were based on the TREC [19] guidelines, which were modified to suite the INEX task [10]. There are 30 CO and 30 CAS topics in the first topics set. CO topics do not have any structural

constraints and are similar to traditional IR queries. Details about the CO topics can be found in [9] and [1]. In CAS topics the user can restrict the context of interest or context of certain search words by explicitly confining the search words to a structural part of the XML document. CAS topics are of interest to us because they are especially suitable for structured retrieval. In this thesis, we use the 30 INEX 2002 CAS topics to evaluate our approach to structured retrieval.

```
<INEX-Topic topic-id="08" query-type="CAS" ct-no="047">
  <Title>
    <te>article</te>
    <cw>ibm</cw><ce>fm/aff</ce>
    <cw>certificates</cw><ce>bdy/sec</ce>
  </Title>
  <Description>
    Find all articles that deal with digital
    certificates and in particular on using certificates
    for authentication purpose.
  </Description>
  <Narrative>
    Relevant documents should deal with solutions that
    use certificates for authenticating users on the
    internet. We are looking only for work that were
    done by IBM.
  </Narrative>
  <Keywords>
    ibm, internet, public-key certificates, security
  </Keywords>
</INEX-Topic>
```

Figure 2: A CAS topic from the INEX test collection

The four main parts of the topic are topic title, topic description, narrative and keywords. The topic title is a short version of the query description and usually consists of keywords that best describe the user's informational need. In addition the topic title describes both content and structure-related requirements. Hence the topic title may contain different components that express this need. These components are: target elements (<te>), a set of search concepts (<cw>), and a set of context elements for the

search concepts (<ce>). The combination of the later two corresponds to a containment condition. A search concept may be represented by a set of keywords or phrases. Both target and context elements may list one or more XML elements (e.g., <ce>abs, kwd</ce>), which may be given by their absolute (e.g., article/fm/au) or abbreviated path (e.g., / /au) or by their element type (e.g., au).

The target element specifies the type of XML element the search should return. In the CAS topic shown in Figure 2, the target element <te> is “article” which means that the search should return a list of *article elements*. The context element specifies the type of XML element or elements within which a given search concept should be searched. In the CAS topic shown in Figure 2, the context element for the search concept “ibm” is the element <aff>. The <aff> element gives the affiliation of the authors of the article. So this containment condition states that “ibm” should be contained within or should be the subject of the <aff> element. What the user is trying to specify here is that he is interested in only those articles whose authors are affiliated with IBM. Similarly, the second containment condition states that the search word “certificates” should be the subject of the element <sec>. We can say that the user is interested in articles written by authors affiliated with IBM that deal with “certificates”. The narrative of the topic confirms this informational need of the user.

Omitting the target element or the context element in a topic title indicates that there are no restrictions placed upon the type of element the search should return, or on the type of element of which a given concept should be a subject.

The query description is a one or two sentence, natural language definition of the information need. The narrative is a detailed explanation of the query statement and a description of what makes a document or a document component relevant or not. The keywords component of the query gives a list of search terms related to the original query that may help in retrieval.

The three attributes of a topic are: topic-id (e.g., 1 to 60), query-type (e.g., CAS or CO), and ct-no, which refers to the candidate topic number (e.g., 1 to 143) assigned to a topic during the development phase.

2.2 Assessments

In the traditional evaluation of information retrieval, e.g., TREC, relevance is judged on a document level, which is the atomic unit of retrieval. In XML retrieval, the retrieval results may contain document components of varying granularity, e.g., *paragraphs*, *sections*, *author names*, *article titles*, etc. This granularity calls for modification of the assessment metrics.

Here we briefly describe the assessment metrics developed at and used by the participants of the INEX 2002 workshop (described at length in [11]). The document components returned as a result of retrieval are assessed for the following two dimensions.

Topical relevance, which describes the extent to which the information contained in a document component is relevant to the topic of request.

Document coverage, which describes how much of the document component is relevant to the topic of request.

To assess the topical relevance dimension, the following 4-point relevance degree scale was adopted.

- 0: Irrelevant, the document component does not contain any information about the topic of the request.
- 1: Marginally relevant, the document component mentions the topic of the request but only in passing.
- 2: Fairly relevant, the document component contains more information than the topic description, but this information is not exhaustive. In the case

of multi-faceted topics, only some of the sub-themes or viewpoints are discussed.

- 3: Highly relevant, the document component discusses the topic of the request exhaustively. In the case of multi-faceted topics, all or most sub-themes or viewpoints are discussed.

To assess document coverage, the following four categories were adopted.

- N: No coverage, the topic or an aspect of the topic is not a theme of the document component.
- L: Too large, the topic or an aspect of the topic is only a minor theme of the document component.
- S: Too small, the topic or an aspect of the topic is the main or only theme of the document component, but the component is too small to act as a meaningful unit of information when retrieved by itself (e.g., without any context).
- E: Exact coverage, the topic or an aspect of the topic is the main or only theme of the document component, and the component acts as a meaningful unit of information when retrieved by itself.

These two assessment dimensions are not perfectly orthogonal to each other, which means that some combinations of relevance and coverage do not make sense. A document component with no relevance cannot have any coverage of the topic and vice versa. A document component with coverage too small cannot be highly relevant, as highly relevant would assume that all or most of the concepts requested in the topic are discussed exhaustively in the document component.

2.2.1 Implicit Assessments

Due to the nature of the two assessed dimensions one can, in certain cases, deduce assessments for nodes which have not been assessed explicitly.

According to the definition of the relevance dimension, the relevance of a parent component of an assessed component is equal to or greater than the relevance of the assessed component. For a component that has a coverage assessment of *exact* or *too large* it can be deduced that its parent component has coverage of *too large*.

These rules have been applied recursively, up to the article level of the documents, in order to add implicit assessments from the explicit assessments done by the assessors. The only exceptions for applying these rules are the CAS topics with target element specification because the target element specification is interpreted in a strict way in terms of evaluation.

2.3 Evaluation Metrics

The evaluation procedures used in traditional evaluation initiatives like TREC could not be used for the XML retrieval task without modifications, since the nature of the XML task is different from that of the traditional retrieval task. Here we describe the evaluation metrics discussed at the INEX 2002 workshop and applied to the INEX 2002 submissions [9]. We evaluate our approach using these metrics, which are implemented within the `inex_eval` package. The quantisation of the relevance and coverage is discussed in Section 2.3.1, and Section 2.3.2 discusses the recall/precision metrics proposed at INEX 2002.

2.3.1 Quantisation of Relevance and Coverage

Although we have two dimensions for assessment of a XML document component, we need to quantise the relevance and coverage values in order to apply the traditional recall/precision metrics. We need some function f_{quant} that will take as input the relevance and coverage values and give a numerical value as output.

$$f_{quant}: Relevance \times Coverage \rightarrow [0,1]$$

$$(rel, cov) \mapsto f_{quant}(rel, cov)$$

Here, *relevance* can take values $\{0, 1, 2, 3\}$, and *coverage* can take values $\{N, S, L, E\}$.

The quantisation function can be selected to reflect the user's standpoint on relevance and coverage. INEX 2002 uses two quantisation functions, f_{strict} and $f_{generalised}$.

The quantisation function f_{strict} is used to evaluate the capability of a given retrieval method to retrieve highly relevant and exact document components.

$$f_{strict}(rel, cov) = \begin{cases} 1, & \text{if } rel = 3 \text{ and } cov = E \\ 0, & \text{else} \end{cases}$$

The quantisation function $f_{generalised}$ is based on the different possible combinations of relevance degrees and coverage categories. It credits the document components according to their degrees of relevance.

$$f_{generalised}(rel, cov) = \begin{cases} 1.00 & \text{if } (rel, cov) = 3E \\ 0.75 & \text{if } (rel, cov) \in \{2E, 3L\}, \\ 0.50 & \text{if } (rel, cov) \in \{1E, 2L, 2S\}, \\ 0.25 & \text{if } (rel, cov) \in \{1S, 1L\}, \\ 0.00 & \text{if } (rel, cov) = 0N \end{cases}$$

2.3.2 Recall/Precision Metrics

Using the quantisation functions described above, each document component in a result ranking is assigned a numeric relevance value. Thus procedures that calculate the recall/precision curves for traditional document retrieval can be applied directly to the results of the quantisation functions.

One issue to consider before applying the standard evaluation procedures is that of the overlaps of document components in the rankings. This issue is peculiar to XML retrieval because the atomic unit of retrieval is not a single document. So more than one component of the same document can show up in the results. For example, a document

itself (e.g., <article>) and a section of the same document (e.g., <article>/<bdy>/<sec>) may show up in the results, generated in response to a query. In INEX 2002, overlaps of the document components in rankings were ignored [9].

INEX 2002 used the method described by Raghavan *et. al.* [15] for calculating recall/precision curves. We briefly review the procedure as described in [9]. In this method, precision is interpreted as the probability, $P(rel|ret)$, that a document viewed by a user is relevant. Given that the user stops viewing the ranking after a given number of relevant document components NR, this probability can be computed as follows.

$$P(rel | retr)(NR) = \frac{NR}{NR + esl_{NR}} = \frac{NR}{NR + j + s \cdot i / (r + 1)}$$

Here esl_{NR} is the expected search length, which denotes the total number of non-relevant document components that are estimated to be retrieved until the NR^{th} relevant document is retrieved. Let l denote the rank from which the NR^{th} relevant document component is drawn. Then j is the number of non-relevant document components from the ranks before l , s is the number of relevant components to be taken from rank l , and r and i are the numbers of relevant and non-relevant components in rank l . (Details of derivation are given by Cooper in [2].)

Raghavan *et. al.* also give a theoretical justification for using intermediary real numbers instead of simple recall points only.

$$P(rel | ret)(x) = \frac{x \cdot n}{x \cdot n + esl_{x \cdot n}} = \frac{x \cdot n}{x \cdot n + j + s \cdot i / (r + 1)}$$

Here, n is the total number of relevant document components with regard to the user request in the collection and $x \in [0,1]$ denotes an arbitrary recall value.

This gives us an intuitive method for employing arbitrary fractional numbers, x , as recall values and thus allows for averaging evaluation results over multiple topic results [9].

The main advantage of this metric of Raghavan et al. is that the variables n , j , i , r , and s in Formula 5 can be interpreted as expectations, thus allowing for a straightforward implementation of the metric for the generalized quantisation function. For example, given a function $\text{assessments}(c)$, which yields the relevance / coverage assessment for a given document component c , the number n of relevant document components with respect to a given topic and quantisation function is computed as:

$$n = \sum_{c \in \text{components}} f_{\text{quan}}(\text{assessment}(c))$$

Expectations for the other variables are computed respectively [9].

The method Raghavan, *et. al.*, use to compute the recall/precision curves assumes that the submission conceptually ranks all components available through the document collection. However, the participants of INEX 2002 were allowed to report only the top 100 document components per topic. The evaluation procedure therefore creates a virtual final rank, which enumerates all the components not being part of the set of components explicitly ranked within submission itself. A theoretical problem which arises in the case of structured document retrieval is the question of the size of this rank. Considering the fact that not every element given by the XML markup of the documents is a candidate for retrievable component, we estimate this figure (details in [9]). The estimated number of retrievable components for a given topic can then be computed by:

$$|\text{components}| \approx |\text{documents}| \cdot \frac{|\text{components assessed}|}{|\text{documents assessed}|}$$

These evaluation procedures are implemented within the `inex_eval` package freely distributed to the INEX 2002 participants.

3 Adapting the Extended Vector Space Model to CAS Queries

3.1 Vector Space Model

The vector space model [17] is one of the advanced retrieval models. In the vector space model, each document is indexed so as to represent it in the form of a weighted term vector. The indexing procedure uses a stop list to remove commonly occurring words (*and, or, the, of, etc.*) and is followed by a stemming procedure to generate word stems from the remaining words (e.g., *analysis, analyzer* and *analyzing* are reduced to stem *analy*). Next, a weight is assigned to each word stem and each document is represented as a vector of weighted word stems. Thus, the vector space model represents the document collection as a vector space of dimension $m \times n$, where m is the number of unique word stems in the document collection and n is the number of documents in the collection. The queries, like documents, are represented as weighted term vectors.

The weight assigned to a particular term in a document vector is indicative of the contribution of the term to the meaning of the document. Most of the weighting schemes are based on *tf-idf* weighting, in which each term is assigned a weight equal to the product of its term frequency (the number of times the term occurs in the document) and a function of its inverse document frequency (the total number of documents in the collection in which the term appears). The *tf* factor reflects the importance of a term within the document and the *idf* factor reflects the importance of the term within the document collection. The higher the term frequency, the more important the term is within the document. The higher the document frequency of a term, the less important the term is within the document collection. So we use inverse document frequency.

In the vector space model, the relevance of a document to a query is represented by the mathematical similarity of their corresponding term vectors. A commonly used measure of the similarity of vectors is the cosine similarity measure, the cosine of the angle

between the two vectors. The smaller the angle between the corresponding vectors, the greater the similarity of the two vectors [17].

Thus, the vector space model gives us a simple yet effective retrieval model. Figure 3 (adapted from [5]) shows an example of the vector space model.

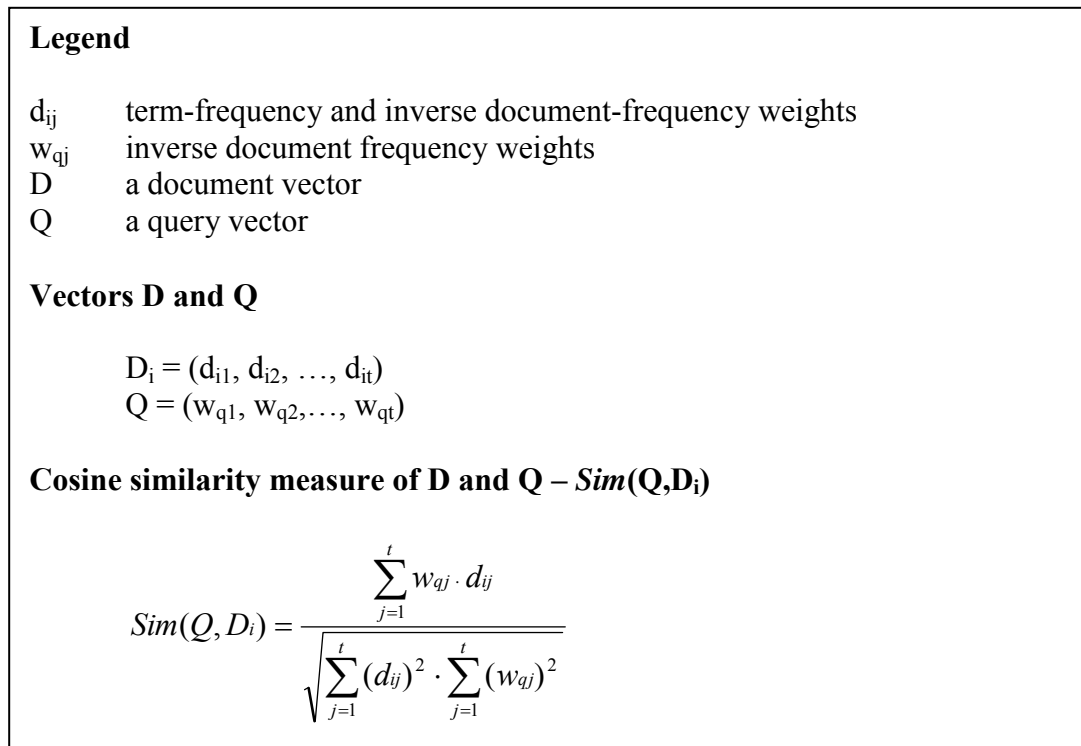


Figure 3: An Example of Vector Space Model

3.2 Extended Vector Space Model

Fox extended the vector space model [7] to include concepts other than the normal content terms. He developed a method for representing in a single, extended vector different classes of information about a document, such as author name, terms, bibliographic citations, etc. In the extended vector space model, a document vector consists of a set of subvectors, where each subvector represents a different concept type

or c-type. Similarity between a pair of extended vectors is calculated as a linear combination of the similarities of corresponding subvectors. Figure 4 (adapted from [5]) contains an example of the extended vector space concept.

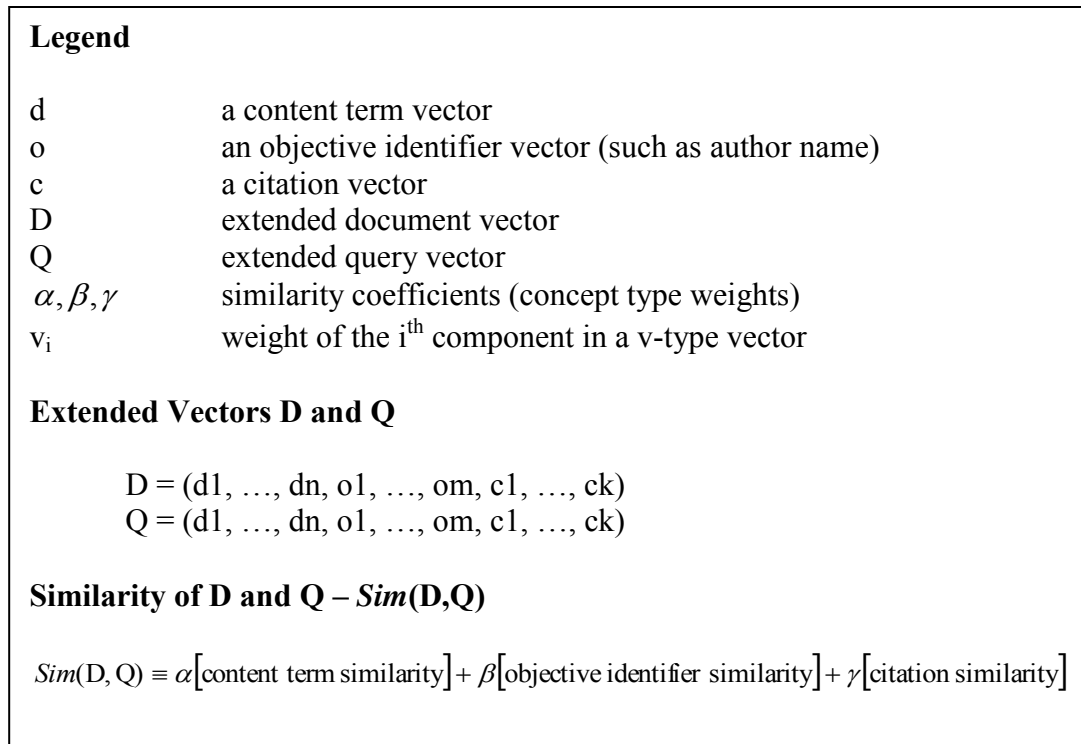


Figure 4: An Example of an Extended Vector

The extended vector space model is suitable for representing documents with different classes of information. It facilitates formulation of search strategies which take advantage of particular combinations of concept types.

3.3 Adapting the Extended Vector Space Model to CAS Queries

In this section we describe our approach to the structured retrieval task. We apply the extended vector space model to CAS queries with some modifications, thus adapting it to the structured retrieval task.

It is evident from the description of the documents and CAS queries in Chapter 2 that they contain multiple concept classes. We identified 18 different concept types or c-types. Table 2 lists the c-types along with a brief description of each c-type.

Table 2: C-types and their description

c-type	Description
abs	Abstract
ack	Acknowledgements
article_au_fnm	First name of article's author
article_au_snm	Surname of article's author
atl	Article title
au_aff	Author affiliation
bibl_atl	Article title in bibliography
bibl_au_fnm	Author's first name in bibliography
bibl_au_snm	Author's surname in bibliography
bibl_ti	Publication (journal) title in bibliography
ed_aff	Editor affiliation
ed_intro	Editor Introduction
kwd	Keywords
rname	Reviewer name
st	Section title
ti	Publication (journal) Title
pub_yr	Publication year
words	Text from Article body/section/paragraphs

We could not directly apply the extended vector space model to the CAS queries. We split certain CAS queries into separate portions which are then run as two separate queries, and the results are combined in a specific fashion to ensure that the elements retrieved meet the specific criteria.

Consider, for example, the title section of CAS query 8:

```
<Title>
  <te>article</te>
  <cw>ibm</cw><ce>fm/aff</ce>
  <cw>certificates</cw><ce>bdy/sec</ce>
</Title>
```

In this case, the query is to return a list of articles as specified by the target element `<te>`. The narrative of the query specifies that sections of relevant documents should contain information about the use of certificates for authenticating users on the internet. And since the context of the word *ibm* is *fm/aff*, the author(s) of those documents must be affiliated with IBM. Thus the query should retrieve only those articles on the use of certificates whose author(s) are affiliated with IBM [3].

Direct use of the extended vector model does not guarantee that each keyword will occur in the specified context. The extended vector model returns a ranked list of documents descending in the similarity value. It is evident from the similarity calculations (described in Section 3.2) that a document containing the highly weighted term *certificates* in its sections and whose author(s) are not affiliated with IBM (i.e., the document does not contain the term *ibm* in *fm/aff*) will show up in the ranked list. We deal with this issue by splitting the query into two queries as follows:

Query-1: `<cw>ibm</cw><ce>fm/aff</ce>`

Query-2: `<cw>certificates</cw><ce>bdy/sec</ce>`

Author affiliation and section are two different c-types. So Query-1 searches for documents containing the objective identifier *ibm* in the author affiliation subvector. Query-2 seeks documents whose section(s) contain the subjective identifier *certificates*. Our retrieval system returns a ranked list of documents for both queries. The intersection of these lists is the final, ranked list of documents returned for query 8. The ranking in the final list assumes the ranking for the subjective part of the query because the ranking for objective part of the query does not have any meaning. The list of

documents returned by the objective query gives us a subset of documents from the document collection satisfying the objective condition, but the ranking has no significance as far as relevance of the subjective part is concerned. For other CAS queries, the results of the split queries are combined using appropriate set operations, e.g., intersection, union, etc.

We represent the documents and the modified set of CAS queries in extended vector form. The extended vector itself is a combination of subvectors, some containing normal text and others containing objective identifiers associated with the document. We use the Smart experimental retrieval system [16], which implements the vector space model and its extension to accommodate different c-types as proposed by Fox. The Smart system evolved over a 30-year period under the direction of Gerard Salton and Chris Buckley and other researchers at Cornell University.

We perform the following steps for structured retrieval:

1. The XML documents are pre-parsed using a simple XML parser. This results in a parsing of the collection such that each of our 18 c-types is now identifiable in terms of its XML path.
2. The documents and modified split queries are translated into Smart format. Smart indexes these documents and queries as extended vectors.
3. Retrieval takes place by running the queries against the indexed document collection with subvector-to-subvector matching. The result is a list of documents ordered by decreasing similarity to the query. (A variety of weighting schemes is available through Smart.)
4. The results from the split queries are merged in a manner specific to that CAS query. The final results are automatically converted to INEX submission format and a ranked list of target elements is reported.

3.4 Implementation Details

In this section we describe the implementation details of our system. Section 3.4.1 describes the pre-parser implementation, Section 3.4.2 describes the implementation of

the result merger (i.e., query resolution), and Section 3.4.3 describes the conversion of results from Smart to INEX format.

3.4.1 The Pre-parser

The documents from the INEX 2002 test collection are in XML format. We pre-parse these documents such that each of our 18 c-types are identifiable in terms of its XML path. For indexing the documents using Smart the documents must be in the appropriate format. Our pre-parsing also ensures that the documents are in Smart-acceptable format.

We use the Expat XML parser which is freely available on the web at [6]. Expat is a library, written in C, for parsing XML documents. It is the underlying XML parser for the open source Mozilla project, Perl's XML::parser, and other open source XML parsers. It is very fast and also sets high standards for reliability, robustness and correctness.

Expat is a stream-oriented parser. Handler functions are declared with the parser and parsing begins by feeding the document to the parser. As the parser recognizes parts of documents, it calls the appropriate handler for that part of the document (if one is declared). For example, if a handler is declared for start tag, the parser will call this handler when a start tag is encountered in the XML document fed to the parser. The document is fed to the parser in pieces, so parsing begins before the whole document is available. It is completely up to the calling application how much of the document to fit into a piece. This allows us to parse huge documents without worrying about the memory constraints. The user can customize the handler functions. A simple do nothing handler can be an empty function. Table 3 lists handlers used and the function of each.

We need to keep track of the current context while walking through the XML document hierarchy. Expat, being a stream-oriented parser, does not remember the current context. For instance, Expat cannot answer a simple question like "What element does this text belong to?" since the parser may have descended into other elements that are children of the current one and has encountered this text on the way out. To counter

this problem, we implement a stack mechanism. The start-tag handler pushes the start-tag (start-element) onto the stack and the end-tag handler pops the tag from the stack. Thus, at any point in time, the stack stores the complete path of the current element. In Figure 5, the first stack shows the stack contents when parser has encountered element <ti>, and the second stack shows the stack contents when <st> is encountered. In both the cases the stack stores the context of current element.

Table 3: Expat parser handlers and their functions

Handler name	Handler Function
<code>start_hndl</code>	Element start handler; called when start of and element is encountered
<code>end_hndl</code>	End Element handler; called when end of and element is encountered
<code>char_hndl</code>	Character data handler; called when character data between start and end of and element is encountered
<code>entity_hndl</code>	Handles external entity references
<code>default_hndl</code>	Default handler; called when no handler is defined for data encountered
<code>comment_hndl</code>	Comment handler; called when comment is encountered

Given the stack, we follow the following steps to pre-parse the documents. The output of the pre-parser is written to the output file.

1. While parsing the document, if we encounter the start-tag of an element from the list of 18 c-types, a flag *set* is assigned a value of to 1. The stack is used to match the path of the current element and path of the elements from the list of 18-ctypes. The name of the c-type, enclosed within brackets (<c-type name>), is printed on a new line in the output file.

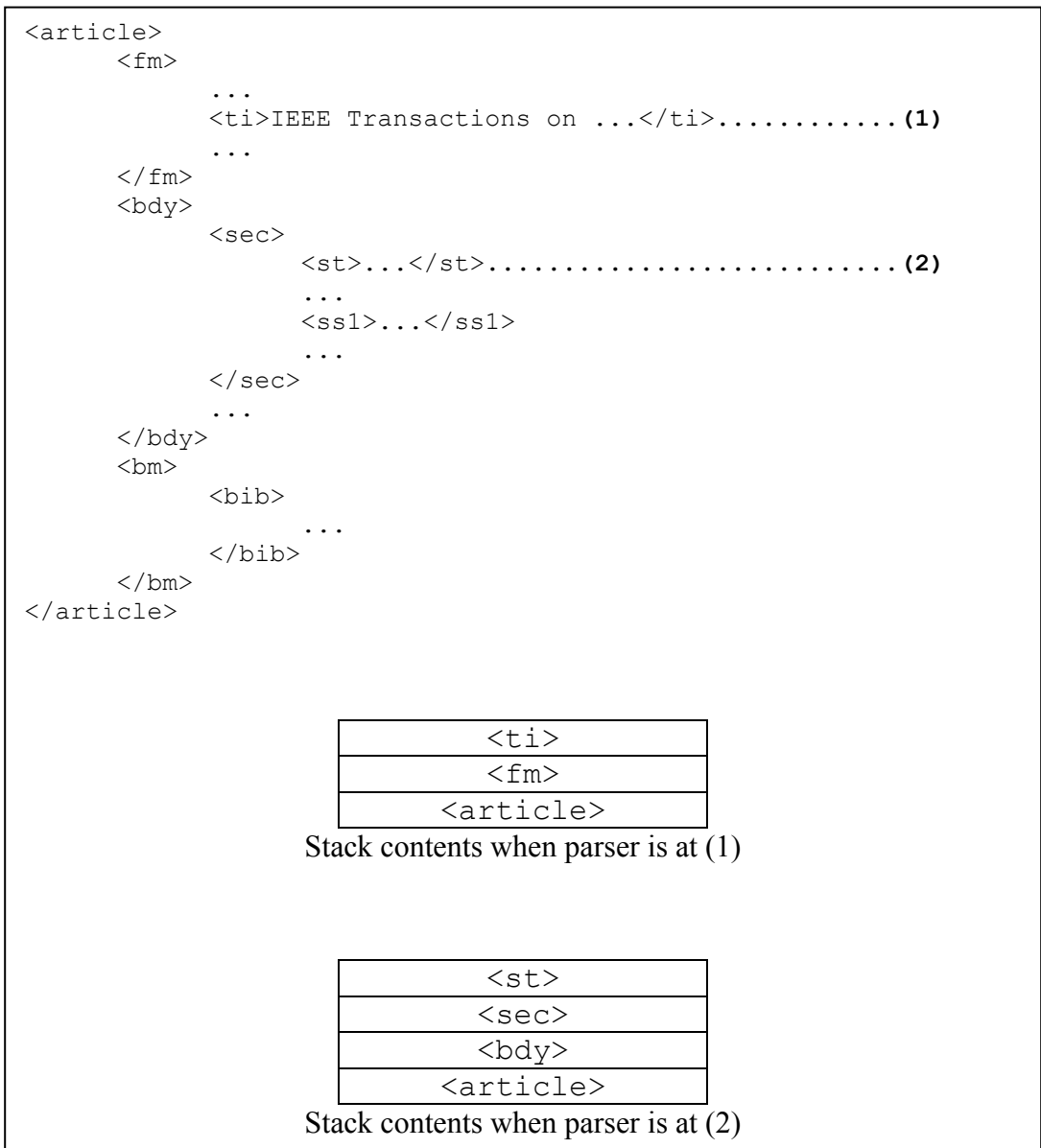


Figure 5: Stack contents at different stages of parsing

2. The text handler checks for the *set* flag. If its value is 1, it copies the text into the output file. The flag *set=1* indicates that the currently encountered text is from the element on the c-types list.

3. When the end-tag is encountered, `set` flag is assigned a value of 0. This ensures that we do not copy the text of an element that is not in our c-types list.

The output file thus produced identifies the text corresponding to all the c-types of interest and is in Smart format.

3.4.2 The Result Merger (Query Resolution)

The result merger module is required to produce a final, ranked list of documents for a CAS query. As we know, each query may be split into two or more queries depending on the original CAS query specifications. These split-queries are run in parallel and Smart retrieves a ranked list of documents for each such split-query. These ranked lists are provided as input to the result merger. Depending on the CAS query specification, the result merger performs AND, OR, NOT operations (or a combination of these operations) on the input lists. The output of the result merger is the final, ranked list of documents. The ranking in the final list follows the ranking for the subjective part of the query. For example, Figure 6 shows the two ranked lists of document numbers for two split-queries and the final list of documents generated after performing the AND operation. The bold-face numbers are the document numbers common to both the split-query lists, and they appear in the final list as a result of intersection. It is evident that the final ranking follows the ranking of the subjective part of the query.

3.4.3 Result Conversion

The INEX evaluation package, `inex_eval`, requires the results in a particular format for evaluation purposes as described in [12]. Figure 7 shows the DTD of the submission format and Figure 8 shows a sample submission.

In the INEX 2002 collection, a document is contained within a single file. File names are relative to the INEX collection's `xml` directory. `,/` is used to separate directories. The extension `.xml` must be left out (e.g., `an/1995/a1004`). Paths are given in XPath syntax. An example path:

```
/article[1]/bdy[1]/sec[1]/p[3]
```

describes the element which can be found if one starts at the document root, selects the first “article” element, then within that element selects the first “bdy” element, within that element selects the first “sec” element, and within that element selects the third “p” element.

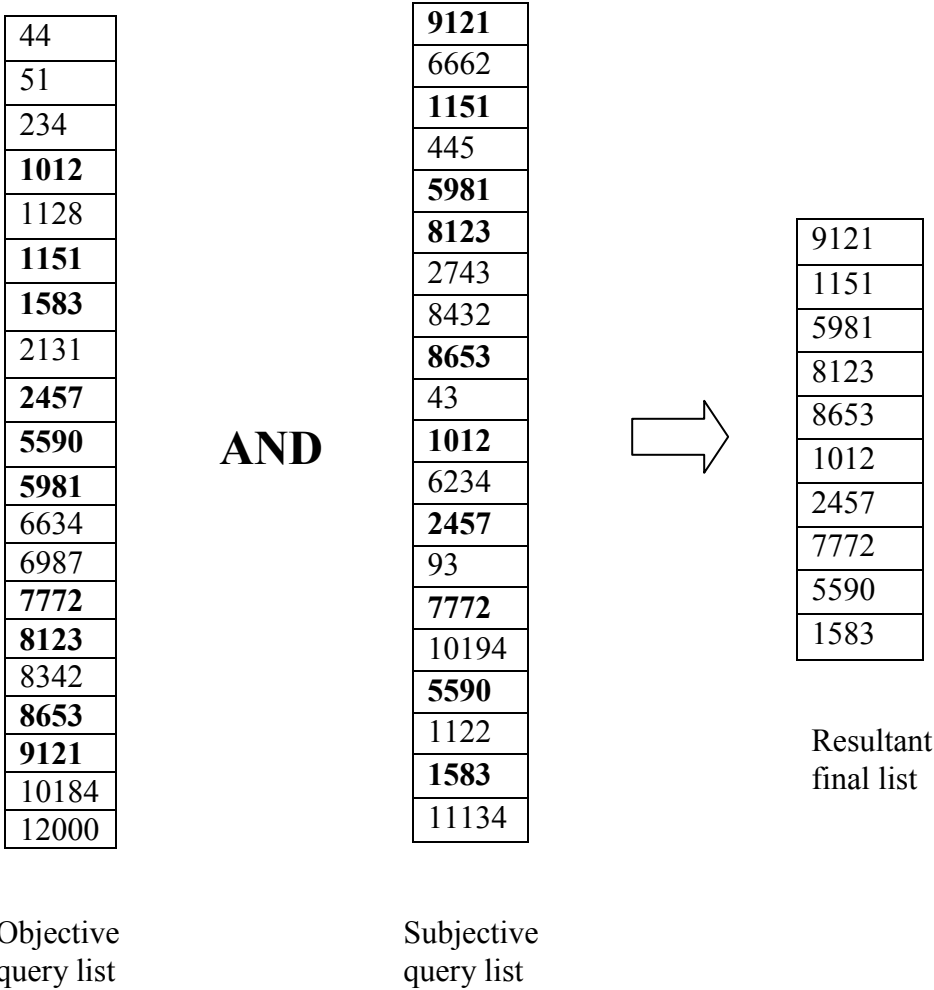


Figure 6: Result merger performs AND operation on the split queries lists to produce a final list. Final ranking follows the ranking of the subjective part of the query.

As can be seen, XPath counts elements starting with one and takes into account the element type. E.g., if a section has a title and 2 paragraphs, then their paths are ./title[1],

../p[1] and ../p[2]. Thus elements are unambiguously identified by a (file name, path) pair.

```
<!ELEMENT inex-submission      (description?, topic+)>
<!ATTLIST inex-submission
    participant-id  CDATA      #REQUIRED
    run-id         CDATA      #REQUIRED
>
<!ELEMENT description  (#PCDATA)>
<!ELEMENT topic        (result*)>
<!ATTLIST topic
    topic-id       CDATA #REQUIRED
>
<!ELEMENT result      (file, path, rank?, rsv?)>
<!ELEMENT file        (#PCDATA)>
<!ELEMENT path        (#PCDATA)>
<!ELEMENT rank        (#PCDATA)>
<!ELEMENT rsv         (#PCDATA)>
```

Figure 7: INEX retrieval result submission format DTD

```
<inex-submission participant-id="12" run-id="MyApproach">
  <topic topic-id="01">
    <result>
      <file>tc/2001/t0111</file>
      <path>/article[1]/bm[1]/ack[1]</path>
      <rank>1</rank>
    </result>
    <result>
      <file>an/1995/a1004</file>
      <path>/article[1]/bm[1]/ack[1]</path>
      <rank>2</rank>
    </result>
    [ ... ]
  </topic>
  <topic topic-id="02">
    [ ... ]
  </topic>
  [ ... ]
</inex-submission>
```

Figure 8: Example INEX retrieval result submission

As described in Section 3.4.2, the result merger module gives us a ranked list of elements. Each element number corresponds to an element of a document in the document collection.

We need the file name within which the document is contained and the XPath of the element within the document. Smart can be customized to keep track of the element number and its corresponding file name. We print the XPath of each c-type (which is some element) to the output file of pre-parsing stage (refer Section 3.4.1), as we also have the XPath of the element we want to report. Our conversion module, written in Perl, extracts the required XPath. Using the file name and the path information, we generate the result file in the specified format.

4 Experiments

In this chapter we discuss the experiments carried out using the INEX data and the extended form of the CAS queries.

4.1 The Setup

This section describes our setup common to all experiments. We use the 30 CAS topics in the INEX 2002 dataset for our experiments. As described in Chapter 2, a CAS topic consists of four parts. We use only the topic title and the keywords while constructing the query. This means we use only the search words provided in the topic title and the keywords part of the topic. Though the topic description and narrative contain useful information, we do not attempt to use that information in query construction. As described earlier, the queries are split wherever required. The constructed queries are represented in Smart format. A total of 51 runs were submitted, by all the participants to INEX 2002, out of these 42 were CAS runs and 49 were CO runs. The results were compared based on the average precision value of each run. The `inex_eval` package calculates the average precision for 100 recall points. As described earlier, INEX uses strict and generalized quantisations for the calculations of average precision. We compare our results against the results of the 42 CAS runs. The top 10 results are readily available on the INEX 2002 up-download area website [13].

4.2 Initial Experiments

Using this setup, we perform our initial run. We use the *Lnu-ltu* weighting scheme [18] for all subvectors. Our earlier work [4] with TREC data shows that the *Lnu-ltu* weighting scheme works best in that environment. Weighting amongst the subvectors is not considered at this point. We weigh all subvectors equally with a weight of 1.0 (which is the default value in Smart). We retrieve the top 100 documents for each query (i.e., for each portion of the split query). The result merger module produces the final list after merging the results from the split queries. We obtain an average precision of 0.103 with generalized quantisation as reported in [3]. This result was not in the top 10 of the 42 results reported at INEX 2002.

After analyzing this initial run, we found the following drawbacks.

1. We retrieved the top 100 documents for the split queries. After merging these results, the list contained fewer than 100 documents as a result of the intersection operation. The initial pool of 100 given to the result merger is too small.
2. The objective queries give us a subset of documents satisfying the objective condition. The window of 100 documents is too small for many objective queries. There are often more than 100 documents satisfying the objective condition. This factor negatively impacts the final results.
3. Our initial system was not tuned to return the results in XPath format as stated in the INEX submission guide [12]. So some of the query results contained invalid XPaths and were not considered for average precision calculation.

We rectified these drawbacks for rest of the experiments. We retrieve 1000 documents for each split query. Also, we implemented a module to return the results in XPath format.

4.3 Experiments with Different Term Weighting Schemes

With the drawbacks of the initial run rectified, we experiment with different term weighting schemes. For all these experiments, we keep the weighting amongst the subvectors equal for all subvectors (at 1.0 for all subvectors). All the weighting schemes discussed here are available through Smart system.

4.3.1 *Lnu-ltu* Weighting Scheme

In this set of experiments, we use Amit Singhal's *Lnu-ltu* weights for all subvectors. *Lnu-ltu* weighting scheme has been used successfully for TREC collections. We retrieve the top 1000 documents for each query and merge the results to get the final results. We obtain an average precision of 0.179 under generalized quantisation and 0.222 under strict quantisation. Our average precision is between sixth and the seventh best runs for

generalized quantisation and between eighth and ninth for strict precision at INEX 2002.

Table 4: Comparison of our results for variation of *Lnu-ltu* weighting scheme to those reported at INEX 2002 for generalized quantisation

Runs	Average Precision (generalized)
1 st in INEX	0.2752
6 th in INEX	0.2419
<i>nnn</i> weights to objective subvectors and <i>Lnu-ltu</i> weights to subjective subvectors	0.187
<i>Lnu-ltu</i> weights to all subvectors	0.179
7 th in INEX	0.1782
10 th in INEX	0.1583

Table 5: Comparison of our results for variation of *Lnu-ltu* weighting scheme to those reported at INEX 2002 for strict quantisation

Runs	Average Precision (strict)
1 st in INEX	0.3438
6 th in INEX	0.3090
<i>nnn</i> weights to objective subvectors and <i>Lnu-ltu</i> weights to subjective subvectors	0.235
7 th in INEX	0.2257
8 th in INEX	0.2233
<i>Lnu-ltu</i> weights to all subvectors	0.222
9 th in INEX	0.1865
10 th in INEX	0.1839

The objective queries serve the purpose of reducing the search set to a set of documents that satisfy the objective condition. So in the next experiment, we weigh the objective

subvectors (i.e., article author's first name and last name, publication year, author's first and last name in bibliography, editor affiliation) with term frequency (*nnn*) weights. Subjective subvector weights are retained as *Lnu-ltu*. We obtain an average precision of 0.187 under generalized quantisation and 0.235 under strict quantisation. Our average precision is between sixth and the seventh best runs for generalized quantisation and between sixth and seventh for strict precision at INEX 2002. Tables 4 and 5 compare our results with those reported at INEX 2002.

4.3.2 *atc* Weighting Scheme

Lnu-ltu weights are most suitable for a document collection with documents of varying size and favor long documents over short documents because long documents have a greater probability of relevance. Since we divide the documents into subvectors, the size of each subvector is much smaller compared to the document as a whole. Hence we use simple normalized augmented *tf-idf* weights (*atc*) weights.

In the first experiment we weight all the subvectors using *atc* weights. We obtain an average precision of 0.194 under generalized quantisation and 0.238 under strict quantisation. Then we weight the objective subvectors with *nnn* weights and subjective subvectors with *atc* weights. We obtain an average precision of 0.192 under generalized quantisation and the average precision increased to 0.243 under strict quantisation. Finally we weigh the body (<body>) subvector with *Lnu-ltu* weights and rest of the subjective subvectors with *atc* weights. Objective subvectors are weighted with *nnn* weights. The average precision we obtain is 0.169 for generalized quantisation and 0.206 for strict quantisation. We observe that heterogeneous weighting schemes for subjective subvectors do not work well. The similarity value calculations are affected by different weighting schemes. Tables 6 and 7 compare our results with those reported at INEX 2002. Tables 8 and 9 compare our results for different weighting schemes for generalized and strict quantisations.

Table 6: Comparison of our results for variation of *atc* weighting scheme to those reported at INEX 2002 for generalized quantisation

Runs	Average Precision (generalized)
1 st in INEX	0.2752
6 th in INEX	0.2419
<i>atc</i> weights to all subvectors	0.194
<i>nnn</i> weights to objective subvectors and <i>atc</i> weights to subjective subvectors	0.192
7 th in INEX	0.1782
<i>nnn</i> weights to objective subvectors, <i>Lnu-ltu</i> weights to body subvector and <i>atc</i> weights to rest of the subvectors	0.169
10 th in INEX	0.1583

Table 7: Comparison of our results for variation of *atc* weighting scheme to those reported at INEX 2002 for strict quantisation

Runs	Average Precision (strict)
1 st in INEX	0.3438
6 th in INEX	0.3090
<i>nnn</i> weights to objective subvectors and <i>atc</i> weights to subjective subvectors	0.243
<i>atc</i> weights to all subvectors	0.238
7 th in INEX	0.2257
8 th in INEX	0.2233
<i>nnn</i> weights to objective subvectors, <i>Lnu-ltu</i> weights to body subvector and <i>atc</i> weights to rest of the subvectors	0.206
9 th in INEX	0.1865
10 th in INEX	0.1839

Table 8: Comparison of results for different weighting schemes under generalized quantisation

Runs	Average Precision (generalized)
<i>atc</i> weights to all subvectors	0.194
<i>Nnn</i> weights to objective subvectors and <i>atc</i> weights to subjective subvectors	0.192
<i>Nnn</i> weights to objective subvectors and <i>Lnu-ltu</i> weights to subjective subvectors	0.187
<i>Lnu-ltu</i> weights to all subvectors	0.179
<i>nnn</i> weights to objective subvectors, <i>Lnu-ltu</i> weights to body subvector and <i>atc</i> weights to rest of the subvectors	0.169

Table 9: Comparison of results for different weighting schemes under strict quantisation

Runs	Average Precision (strict)
<i>Nnn</i> weights to objective subvectors and <i>atc</i> weights to subjective subvectors	0.243
<i>atc</i> weights to all subvectors	0.238
<i>Nnn</i> weights to objective subvectors and <i>Lnu-ltu</i> weights to subjective subvectors	0.235
<i>Lnu-ltu</i> weights to all subvectors	0.222
<i>nnn</i> weights to objective subvectors, <i>Lnu-ltu</i> weights to body subvector and <i>atc</i> weights to rest of the subvectors	0.206

4.4 Weighting amongst the Subvectors

For all the experiments described in Section 4.3, we do not consider weighting amongst the subvectors. All the subvectors are assigned equal weights (1.0). We can assign different concept class weights to subvectors. Let us look at the formula to calculate the similarity value for extended vector space model.

$$Sim(D, Q) \equiv \alpha[\text{similarity}_{\text{subvec1}}] + \beta[\text{similarity}_{\text{subvec2}}] + \gamma[\text{similarity}_{\text{subvec3}}] + \dots + \delta[\text{similarity}_{\text{subvecN}}]$$

Similarity is the linear weighted sum of the individual subvector similarities. We can assign different values to α, β, γ and so on. This changes the relative contribution from an individual subvector. We experimented with assigning different concept class weights to subjective subvectors. We identified four subjective subvectors, namely, *abstract*, *article title*, *keywords* and *body*. We assigned different weights to those subvectors to study the relative contribution of each subvector towards the final similarity value. We used *atc* weighting scheme for all the subvectors for these experiments, as it gave us the best results.

In the first experiment, we fix the weight of the body subvector to default 1.0 and consistently varied the weights of other subjective subvectors. Table 10 shows the results of this experiment. We observe that as we increase the weights on *abstract*, *article title*, and *keywords* subvectors the average precision continues to decrease for strict and generalized quantisation. We conclude that changing the weights on abstract, article title and keywords subvectors does not improve the average precision. Also, the body subvector contributes the most amongst all the subjective subvectors.

In the second experiment, we increase the weight on the body subvector to 4.0. Of the other three subjective subvectors, we keep the weights of the two subvectors constant (at 1.0) and varied the weight of one subvector (to 0.2). The average precision increases when we weigh the body subvector high and decrease the weights on other three subjective subvectors. The highest average precision obtained is 0.197 for generalized

quantisation with the weight of article title set to 0.2, abstract and keywords set to 1.0 and weight of the body subvector set to 4.0. Table 11 gives the results of this experiment. The average precision in both quantisations does not vary much with these weightings, the lowest being 0.195 and highest being 0.197 for generalized quantisation.

Table 10: Effect of changing weights of abstract, article title, and keywords subvectors by keeping the weight of body subvector constant at 1.0

Abstract subvector weights	Article title subvector weights	Keywords subvector weights	Body subvector weights	Average precision (generalized)	Average precision (strict)
0.5	0.5	0.5	1.0	0.187	0.228
2.0	2.0	2.0	1.0	0.178	0.22
3.0	3.0	3.0	1.0	0.174	0.216

In the third experiment, we keep the weight of the body subvector to 4.0. Of the other three subjective subvectors, we keep the weights of the two subvectors constant (at 0.2) and varied the weight of one subvector (to 1.0). In all the cases we obtain an average precision of 0.128 for generalized quantisation. Table 12 shows the results of this experiment.

Finally we perform some more experiments with weighting amongst the subvectors. We increase the weight on the body subvector and decrease the weight on the other three subvectors. Table 13 lists the average precision values for different weightings on subvectors. The average precision does not vary much and stays in the range of 0.192 to 0.196 for generalized quantisation. For strict quantisation it varies from 0.219 to 0.229.

Table 11: Effect of changing weight of abstract, article title, and keywords subvectors one by one to lower value and keeping the weight of body subvector constant at 4.0

Abstract subvector weights	Article title subvector weights	Keywords subvector weights	Body subvector weights	Average precision (generalized)	Average precision (strict)
0.2	1.0	1.0	4.0	0.195	0.229
1.0	0.2	1.0	4.0	0.197	0.231
1.0	1.0	0.2	4.0	0.195	0.229

Table 12: Effect of changing weight of abstract, article title, and keywords subvectors one by one to higher value and keeping the weight of body subvector constant at 4.0

Abstract subvector weights	Article title subvector weights	Keywords subvector weights	Body subvector weights	Average precision (generalized)	Average precision (strict)
1.0	0.2	0.2	4.0	0.128	0.171
0.2	1.0	0.2	4.0	0.128	0.171
0.2	0.2	1.0	4.0	0.128	0.171

Table 13: Results with higher weights for body subvector and lower weights for abstract, article title, and keywords subvectors

Abstract subvector weights	Article title subvector weights	Keywords subvector weights	Body subvector weights	Average precision (generalized)	Average precision (strict)
0.2	0.2	0.2	4.0	0.196	0.227
0.01	0.01	0.01	8.0	0.192	0.219
1.0	1.0	1.0	8.0	0.196	0.229

5 Conclusions and Future Work

We now discuss the conclusions drawn from our work and give some insight into possible future work.

5.1 Conclusions

Our current work in structured retrieval is still in an early stage of development. We adapted the extended vector space model for structured retrieval. In particular we used the content-and-structure (CAS) queries and XML document collection provided by the INEX initiative to test our approach. In CAS queries, the user can restrict the context of interest or the context of certain search words by explicitly confining the search words to a structural part of the XML document. The target element directs the retrieval of a specific part of the document rather than the document itself. Thus we can do structured retrieval using the CAS queries. This was our first attempt to solve the problem of structured retrieval. Yet even at this point, it seems clear that the extended vector space model provides a viable framework for structured retrieval.

We used several weighting schemes and their combinations for CAS queries. Variations of the *atc* weighting scheme for all subvectors works best for the current set of CAS queries. The *atc* weighting of all subvectors works best under generalized quantization, giving an average precision of 0.194. A combination of the *nnn* weighting scheme for objective subvectors and *atc* weighting scheme for subjective subvectors works best under the strict quantization, giving an average precision of 0.243. Our best results rank between the sixth and seventh best reported at INEX 2002 under both quantisations.

We experimented with weighting amongst the subvectors by assigning different concept class weights to subjective subvectors. Weighting amongst the subvectors did not improve the results significantly. The highest precision obtained was 0.197 for generalized quantisation and 0.231 for strict quantisation. This was not a substantial improvement over the average precision values obtained without using a variation of

weighting amongst subvectors. The body subvector contributes the most towards the results.

Our inability to affect average precision by varying weighting amongst the subvectors is not surprising in these circumstances. A traditional system based on the vector space model returns documents ranked by correlation with the query. One could expect to improve say, precision at 20 ([P@20](#)), either by moving relevant documents into the top-ranked set of 20 documents or by improving the ranks of the relevant documents in that set or both. Precision at top 20 ([P@20](#)) is a common measure used to study weighting amongst subvector. At INEX the results are evaluated based on the number of relevant documents in the set of 100 reported. The window here is very large; average precision can only be improved by moving more relevant documents into the reported set [3]. Subvector weighting alone is unlikely to effect changes of this magnitude. To verify this premise, we calculated [P@20](#). We obtained an average precision of 0.118 (with *atc* weighting scheme for all subvectors) when all the subvectors are weighted equally. The average precision jumped to 0.139 when we applied weighting amongst the subvectors that gave us best results for window size of 100. But since INEX evaluation considers results only in terms of the top 100 target elements, we cannot compare these figures with those of other participants.

5.2 Future Work

It will be interesting to adapt relevance feedback to CAS queries. Relevance feedback makes use of the additional search words from relevant and non-relevant documents to improve the search effectiveness. Traditionally documents are assessed only for relevance. Structured retrieval uses two dimensions, relevance and coverage, for assessment purpose. Also there are two quantisations. Generalized quantisation quantifies the relevance and coverage on a 5-point scale. Therefore we need to amend the traditional relevance feedback strategies to take into consideration these two dimensional assessments.

Subvector weighting definitely seems promising for a small window size. This can be further investigated. Currently the query construction is done using only the topic title and the keywords part of the CAS topic. Natural Language Processing and IR techniques can be applied to make use of the topic description and narrative parts of the topic to extract good search words.

6 Bibliography

- [1] Apte, S. Using the Extended Vector Space Model for Content Oriented XML Retrieval. M.S. Thesis, Department of Computer Science, University of Minnesota Duluth, 2003.

- [2] Cooper, W. Expected search length: A single measure of retrieval effectiveness based on weak ordering action of retrieval systems. *Journal of the American Society for Information Science*, 19:30-41, 1968.

- [3] Crouch, C., Apte, S., and Bapat, H. Using the Extended Vector Model for XML Retrieval. In *Proceedings of the First Workshop of the Initiative for the Evaluation of XML Retrieval (INEX)*, (pp. 95-98), Dagstuhl, Germany, 2002.

- [4] Crouch, C., Crouch, D., Chen, Q., and Holtz, S. Improving the retrieval effectiveness of very short queries. *Information Processing and Management*, 38(1): 1-36, 2002.

- [5] Crouch, C., Crouch, D. and Nareddy, K. The automatic generation of extended queries. In *Proceedings of the 13th Annual International ACM SIGIR Conference*, (pp. 369-383), Brussels, 1990.

- [6] Expath XML parser website – <http://sourceforge.net/projects/expat/>

- [7] 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.

- [8] Geva, S. Extreme File Inversion. In *Proceedings of the First Workshop of the Initiative for the Evaluation of XML Retrieval (INEX)*, (pp. 155-161), Dagstuhl, Germany, 2002.
- [9] Gövert, N. and Kazai, G. Overview of the Initiative for the Evaluation of XML retrieval (INEX) 2002. In *Proceedings of the First Workshop of the Initiative for the Evaluation of XML Retrieval (INEX)*, Dagstuhl, Germany, 2002.
- [10] INEX Guidelines for Topic Development. In *Proceedings of the First Workshop of the Initiative for the Evaluation of XML Retrieval (INEX)*, (pp. 178-181), Dagstuhl, Germany, 2002.
- [11] INEX Relevance Assessment Guide. In *Proceedings of the First Workshop of the Initiative for the Evaluation of XML Retrieval (INEX)*, (pp. 184-187), Dagstuhl, Germany, 2002.
- [12] INEX Retrieval Result Submission Format and Procedure. In *Proceedings of the First Workshop of the Initiative for the Evaluation of XML Retrieval (INEX)*, (pp. 182-183), Dagstuhl, Germany, 2002.
- [13] INEX up-download area website – <http://ls6-www.cs.uni-dortmund.de/ir/projects/inex/download/>
- [14] INEX website – <http://qmir.dcs.qmul.ac.uk/inex/>
- [15] Raghavan, V., Bollmann P., and Jung G. A critical investigation of recall and precision as measures of retrieval system performance. *ACM Transactions on Information Systems*, 7(3):205-229, 1989.
- [16] Salton, G., editor. *The SMART Retrieval System – Experiments in Automatic Document Retrieval*, Prentice-Hall, Englewood Cliffs, NJ, 1971.

- [17] Salton, G., Wong, A., and Yang C. A vector space model for automatic indexing. *Communications of the ACM*, v.18 n.11, p.613-620, Novemebr 1975.
- [18] Singhal, A., Salton, G., Mitra, M., and Buckley C. Document Length Normalization. *Information Processing and Management*, 32(5):619-633, 1996.
- [19] TREC website – <http://trec.nist.gov/>
- [20] XPath: XML Path Language – <http://www.w3.org/TR/xpath>
- [21] XQuery: An XML Query Language – <http://www.w3.org/TR/2003/WD-xquery-20030502/>

7 Appendix

7.1 Recall/Precision curves

7.1.1 Recall/Precision curves for our Initial Experiment

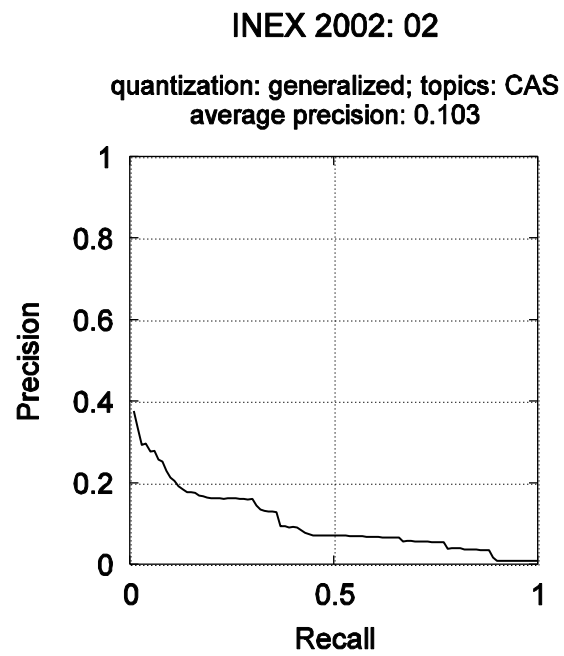


Figure 9: Recall/Precision curve for generalized quantisation – Initial results with *Lnu-ltu* weights to all subvectors

7.1.2 Recall/Precision curves for Experiments with Different Weighting Schemes

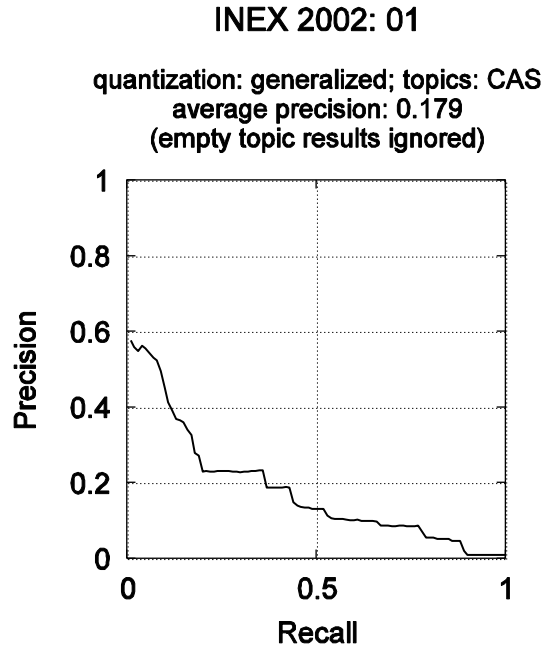


Figure 10: Recall/Precision curves for generalized quantization – *Lnu-ltu* weights to all subvectors

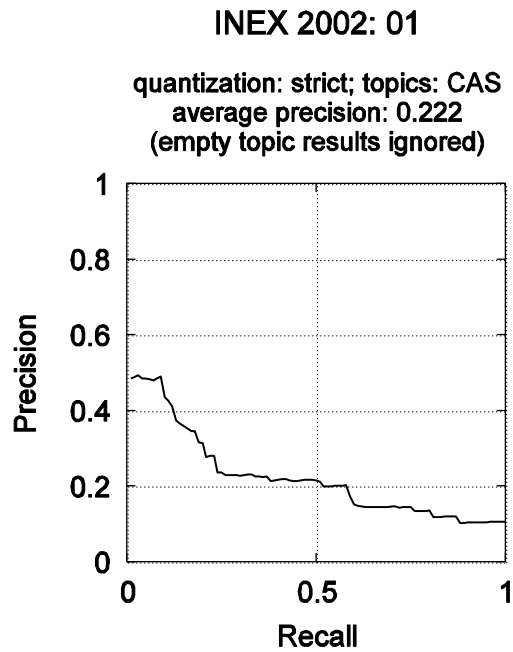


Figure 11: Recall/Precision curves for strict quantisation – *Lnu-ltu* weights to all subvectors

INEX 2002: 01

quantization: generalized; topics: CAS
average precision: 0.187
(empty topic results ignored)



Figure 12: Recall/Precision curves for generalized quantisation – *nnn* weights to objective subvectors and *Lnu-ltu* weights to subjective subvectors

INEX 2002: 01

quantization: strict; topics: CAS
average precision: 0.235
(empty topic results ignored)

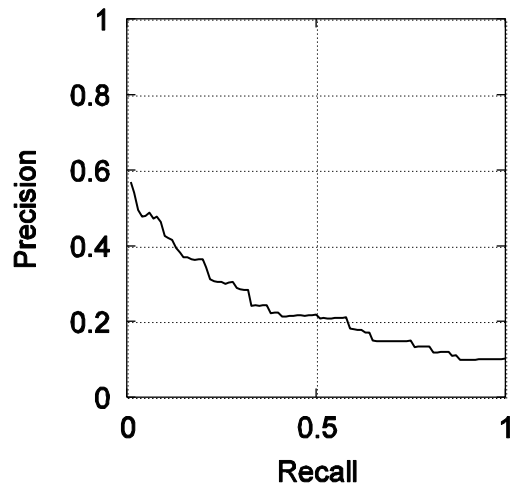


Figure 13: Recall/Precision curves for strict quantisation – *nnn* weights to objective subvectors and *Lnu-ltu* weights to subjective subvectors

INEX 2002: 01

quantization: generalized; topics: CAS
average precision: 0.194
(empty topic results ignored)

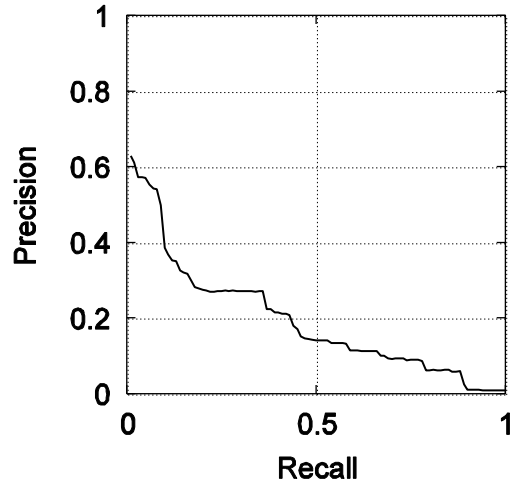


Figure 14: Recall/Precision curve for generalized quantisation – *atc* weights to all subvectors

INEX 2002: 01

quantization: strict; topics: CAS
average precision: 0.238
(empty topic results ignored)

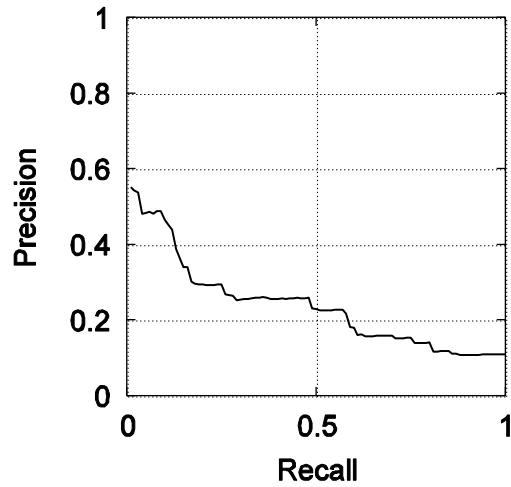


Figure 15: Recall/Precision curve for strict quantisation – *atc* weights to all subvectors

INEX 2002: 01

quantization: generalized; topics: CAS
average precision: 0.192
(empty topic results ignored)

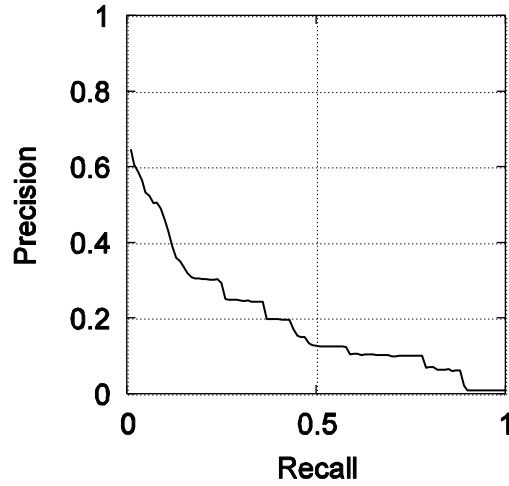


Figure 16: Recall/Precision curve for generalized quantisation – *nnn* weights to objective subvectors and *atc* weights to subjective subvectors

INEX 2002: 01

quantization: strict; topics: CAS
average precision: 0.243
(empty topic results ignored)

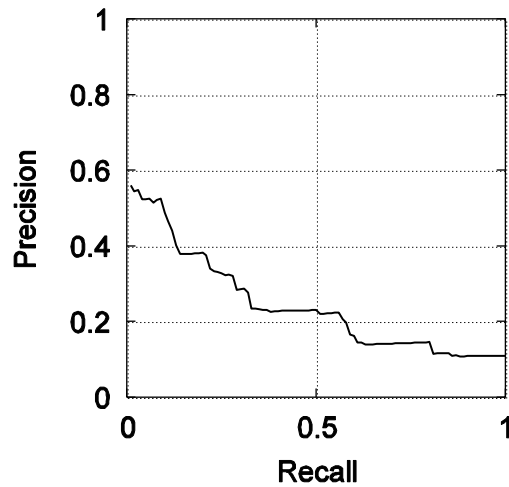


Figure 17: Recall/Precision curve for strict quantisation – *nnn* weights to objective subvectors and *atc* weights to subjective subvectors

INEX 2002: 01

quantization: generalized; topics: CAS
average precision: 0.169
(empty topic results ignored)

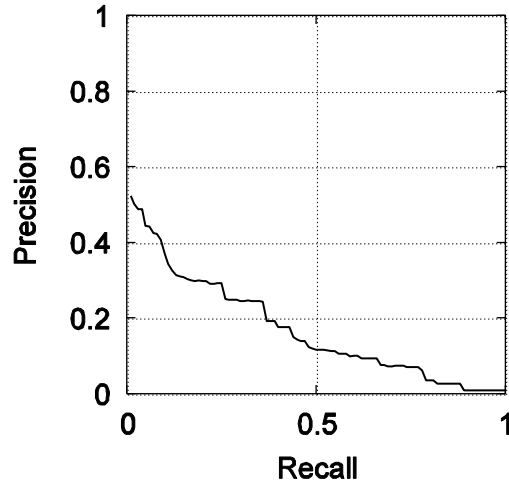


Figure 18: Recall/Precision curve for generalized quantisation – *nnn* weights to objective subvectors, *Lnu-ltu* weights to body subvector and *atc* weights to rest of the subvectors

INEX 2002: 01

quantization: strict; topics: CAS
average precision: 0.206
(empty topic results ignored)

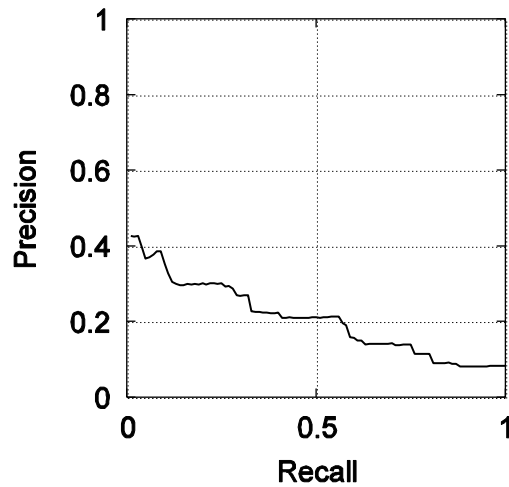


Figure 19: Recall/Precision curve for strict quantisation – *nnn* weights to objective subvectors, *Lnu-ltu* weights to body subvector and *atc* weights to rest of the subvectors

7.1.3 Recall/Precision curves for Weighting amongst the Subvectors

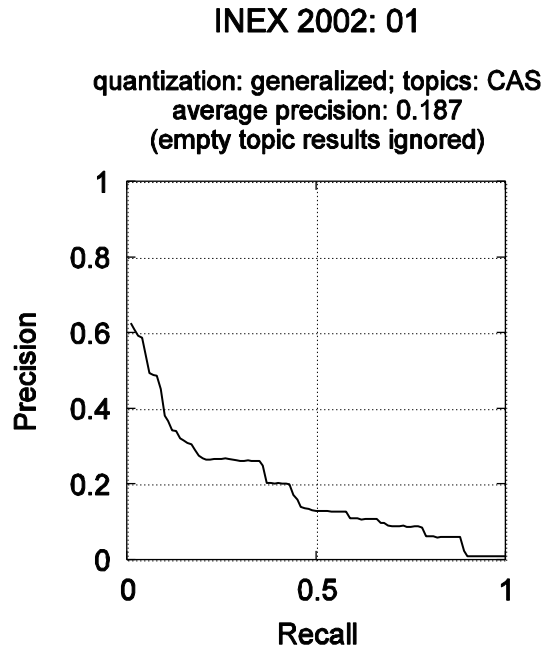


Figure 20: Recall/Precision curve for generalized quantisation with *atc* weights to all subvectors—weighting amongst the subvectors: *abstract=0.5, article title=0.5, keywords=0.5, body=1.0*

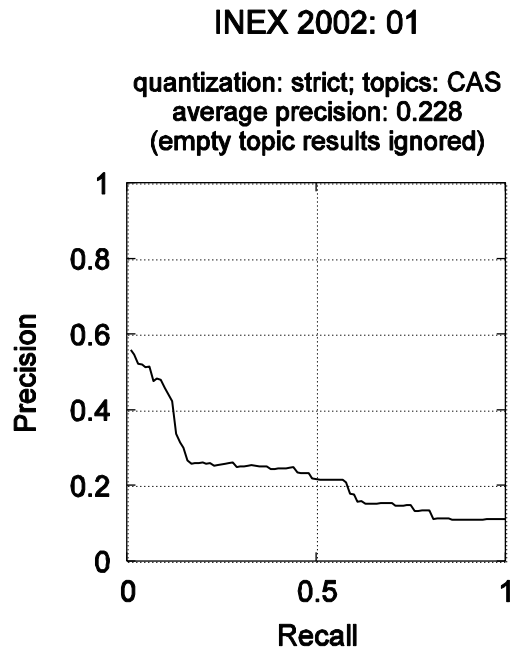


Figure 21: Recall/Precision curve for strict quantisation with *atc* weights to all subvectors—weighting amongst the subvectors: *abstract=0.5, article title=0.5, keywords=0.5, body=1.0*

INEX 2002: 01

quantization: generalized; topics: CAS
average precision: 0.178
(empty topic results ignored)

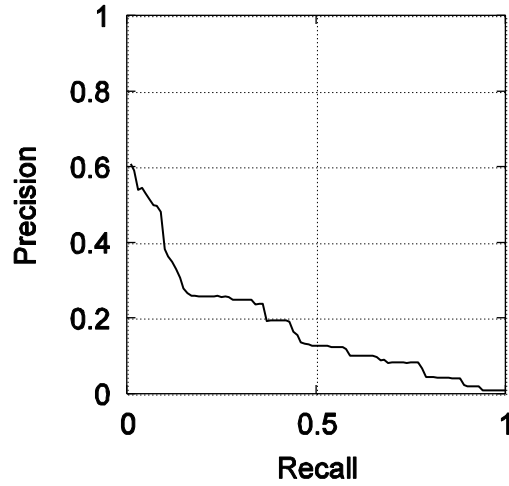


Figure 22: Recall/Precision curve for generalized quantisation with *atc* weights to all subvectors—weighting amongst the subvectors: *abstract=2.0, article title=2.0, keywords=2.0, body=1.0*

INEX 2002: 01

quantization: strict; topics: CAS
average precision: 0.220
(empty topic results ignored)

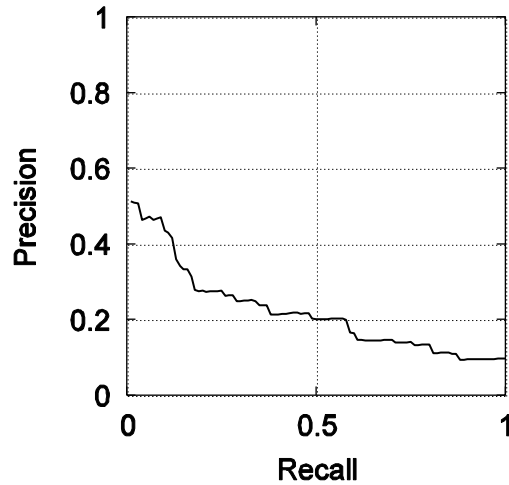


Figure 23: Recall/Precision curve for strict quantisation with *atc* weights to all subvectors—weighting amongst the subvectors: *abstract=2.0, article title=2.0, keywords=2.0, body=1.0*

INEX 2002: 01

quantization: generalized; topics: CAS
average precision: 0.174
(empty topic results ignored)

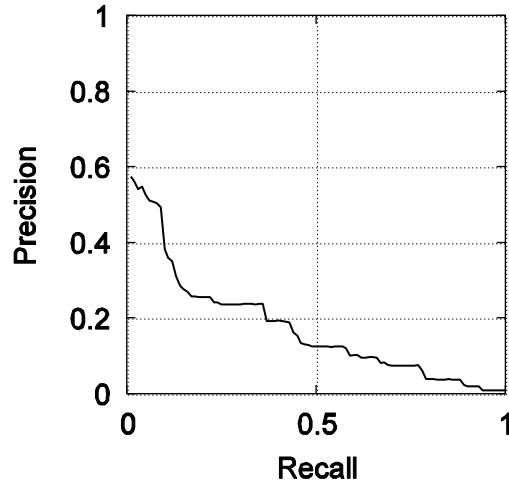


Figure 24: Recall/Precision curve for generalized quantisation with *atc* weights to all subvectors—weighting amongst the subvectors: *abstract=3.0, article title=3.0, keywords=3.0, body=1.0*

INEX 2002: 01

quantization: strict; topics: CAS
average precision: 0.216
(empty topic results ignored)

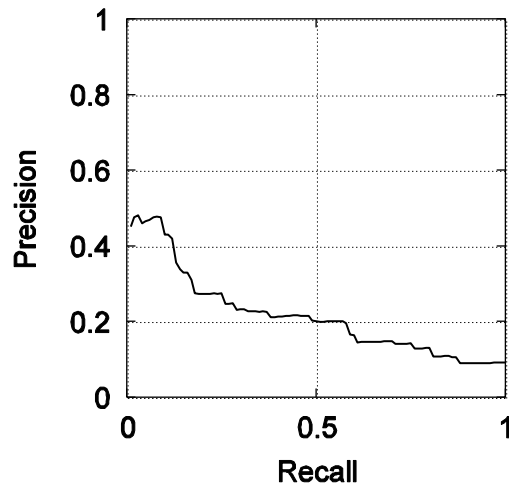


Figure 25: Recall/Precision curve for strict quantisation with *atc* weights to all subvectors—weighting amongst the subvectors: *abstract=3.0, article title=3.0, keywords=3.0, body=1.0*

INEX 2002: 01

quantization: generalized; topics: CAS
average precision: 0.195
(empty topic results ignored)

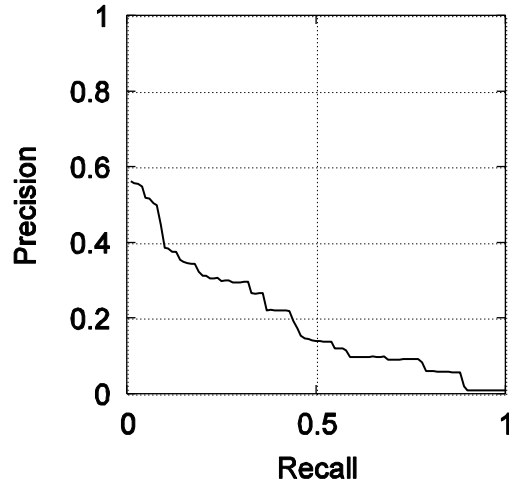


Figure 26: Recall/Precision curve for generalized quantisation with *atc* weights to all subvectors—weighting amongst the subvectors: *abstract=0.2, article title=1.0, keywords=1.0, body=4.0*

INEX 2002: 01

quantization: strict; topics: CAS
average precision: 0.229
(empty topic results ignored)

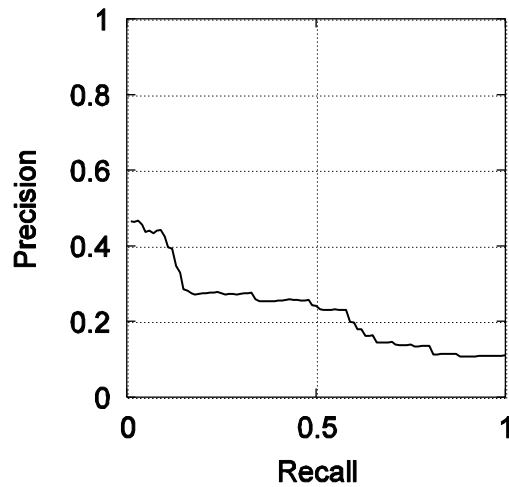


Figure 27: Recall/Precision curve for strict quantisation with *atc* weights to all subvectors—weighting amongst the subvectors: *abstract=0.2, article title=1.0, keywords=1.0, body=4.0*

INEX 2002: 01

quantization: generalized; topics: CAS
average precision: 0.197
(empty topic results ignored)

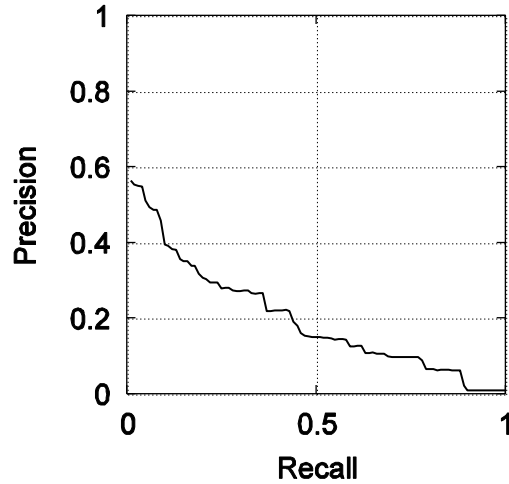


Figure 28: Recall/Precision curve for generalized quantisation with *atc* weights to all subvectors—weighting amongst the subvectors: *abstract=1.0, article title=0.2, keywords=1.0, body=4.0*

INEX 2002: 01

quantization: strict; topics: CAS
average precision: 0.231
(empty topic results ignored)

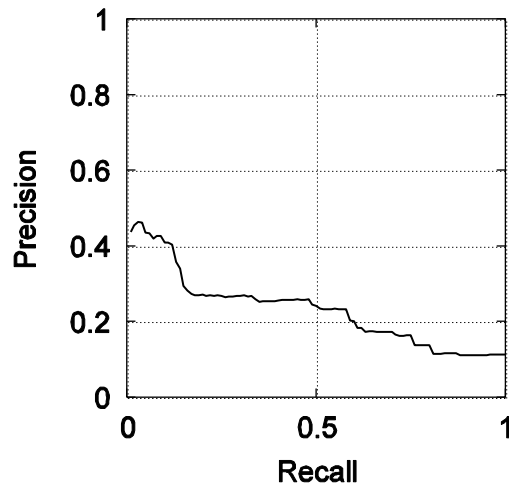


Figure 29: Recall/Precision curve for strict quantisation with *atc* weights to all subvectors—weighting amongst the subvectors: *abstract=1.0, article title=0.2, keywords=1.0, body=4.0*

INEX 2002: 01

quantization: generalized; topics: CAS
average precision: 0.195
(empty topic results ignored)

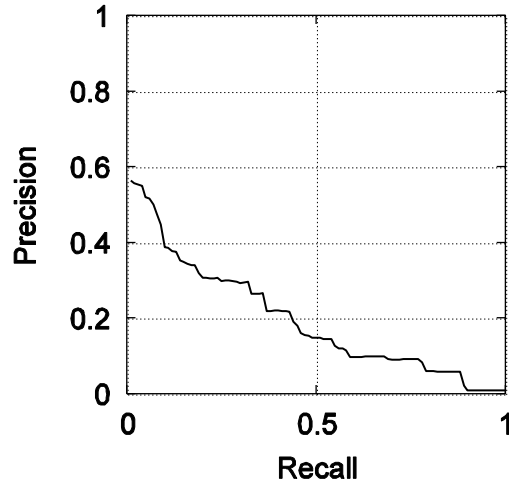


Figure 30: Recall/Precision curve for generalized quantisation with *atc* weights to all subvectors—weighting amongst the subvectors: *abstract=1.0, article title=1.0, keywords=0.2, body=4.0*

INEX 2002: 01

quantization: strict; topics: CAS
average precision: 0.229
(empty topic results ignored)

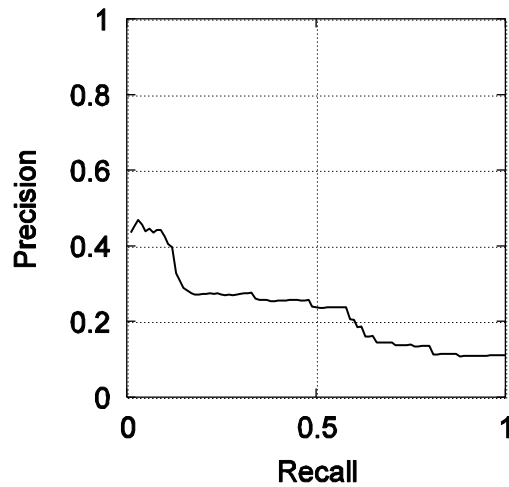


Figure 31: Recall/Precision curve for strict quantisation with *atc* weights to all subvectors—weighting amongst the subvectors: *abstract=1.0, article title=1.0, keywords=0.2, body=4.0*

INEX 2002: 01

quantization: generalized; topics: CAS
average precision: 0.128
(empty topic results ignored)

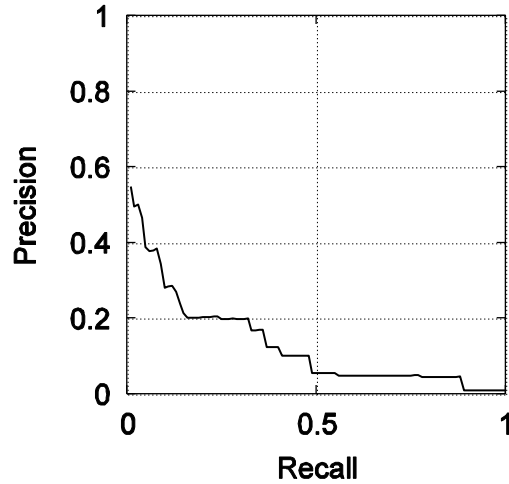


Figure 32: Recall/Precision curve for generalized quantisation with *atc* weights to all subvectors—weighting amongst the subvectors: *abstract=1.0, article title=0.2, keywords=0.2, body=4.0*

INEX 2002: 01

quantization: strict; topics: CAS
average precision: 0.171
(empty topic results ignored)

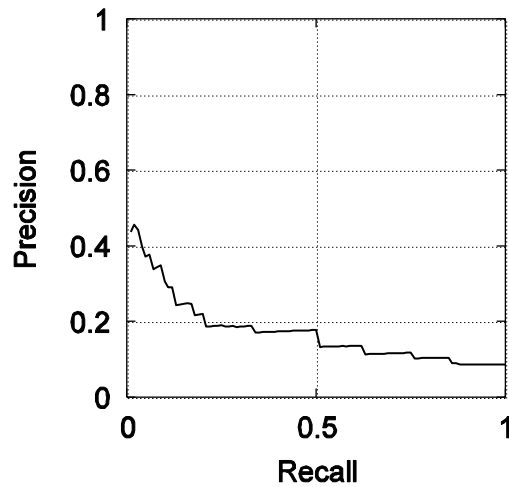


Figure 33: Recall/Precision curve for strict quantisation with *atc* weights to all subvectors—weighting amongst the subvectors: *abstract=1.0, article title=0.2, keywords=0.2, body=4.0*

INEX 2002: 01

quantization: generalized; topics: CAS
average precision: 0.128
(empty topic results ignored)

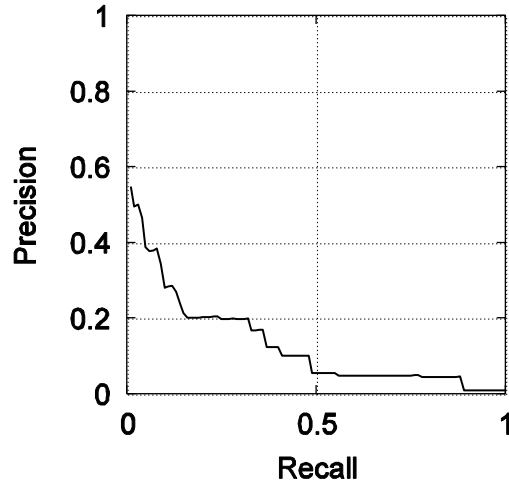


Figure 34: Recall/Precision curve for generalized quantisation with *atc* weights to all subvectors—weighting amongst the subvectors: *abstract=0.2, article title=1.0, keywords=0.2, body=4.0*

INEX 2002: 01

quantization: strict; topics: CAS
average precision: 0.171
(empty topic results ignored)

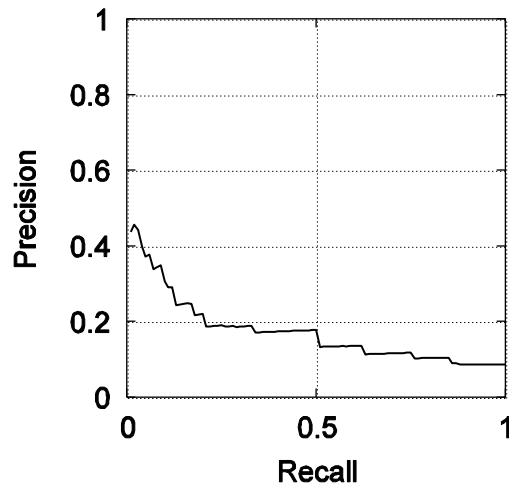


Figure 35: Recall/Precision curve for strict quantisation with *atc* weights to all subvectors—weighting amongst the subvectors: *abstract=0.2, article title=1.0, keywords=0.2, body=4.0*

INEX 2002: 01

quantization: generalized; topics: CAS
average precision: 0.128
(empty topic results ignored)

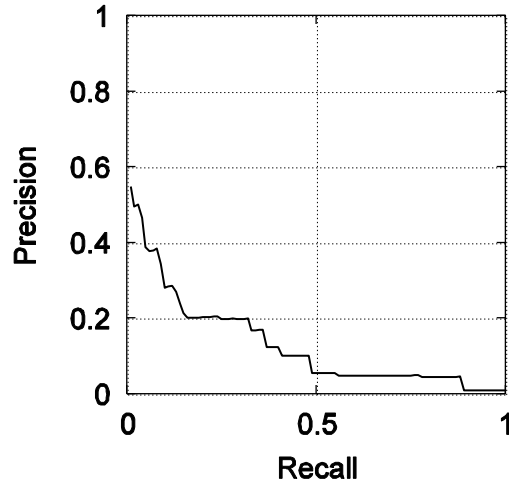


Figure 36: Recall/Precision curve for generalized quantisation with *atc* weights to all subvectors—weighting amongst the subvectors: *abstract=0.2, article title=0.2, keywords=1.0, body=4.0*

INEX 2002: 01

quantization: strict; topics: CAS
average precision: 0.171
(empty topic results ignored)

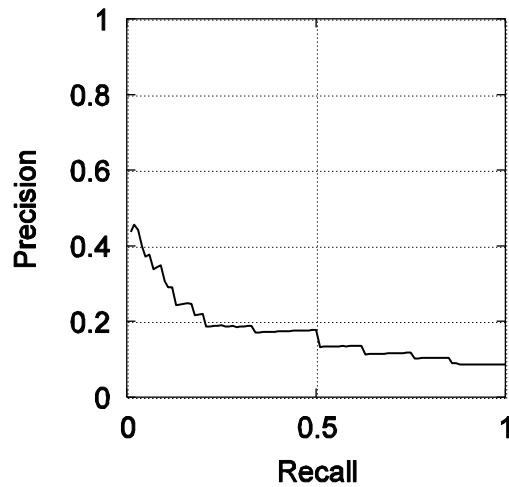


Figure 37: Recall/Precision curve for strict quantisation with *atc* weights to all subvectors—weighting amongst the subvectors: *abstract=0.2, article title=0.2, keywords=1.0, body=4.0*

INEX 2002: 01

quantization: generalized; topics: CAS
average precision: 0.196
(empty topic results ignored)

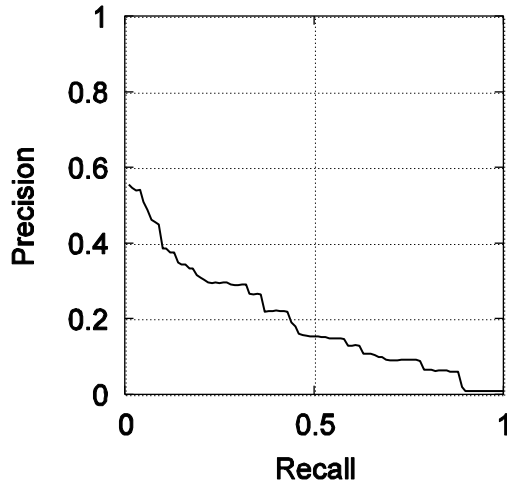


Figure 38: Recall/Precision curve for generalized quantisation with *atc* weights to all subvectors—weighting amongst the subvectors: *abstract=0.2, article title=0.2, keywords=0.2, body=4.0*

INEX 2002: 01

quantization: strict; topics: CAS
average precision: 0.227
(empty topic results ignored)

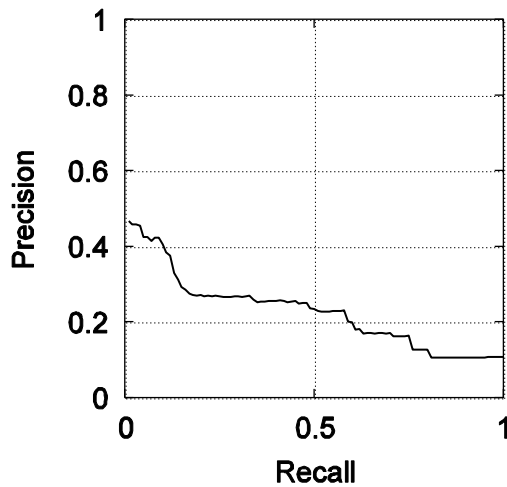


Figure 39: Recall/Precision curve for strict quantisation with *atc* weights to all subvectors—weighting amongst the subvectors: *abstract=0.2, article title=0.2, keywords=0.2, body=4.0*

INEX 2002: 01

quantization: generalized; topics: CAS
average precision: 0.192
(empty topic results ignored)

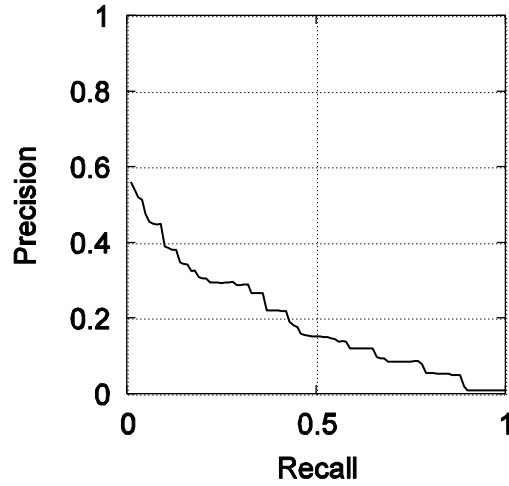


Figure 40: Recall/Precision curve for generalized quantisation with *atc* weights to all subvectors—weighting amongst the subvectors: *abstract=0.01, article title=0.01, keywords=0.01, body=8.0*

INEX 2002: 01

quantization: strict; topics: CAS
average precision: 0.219
(empty topic results ignored)

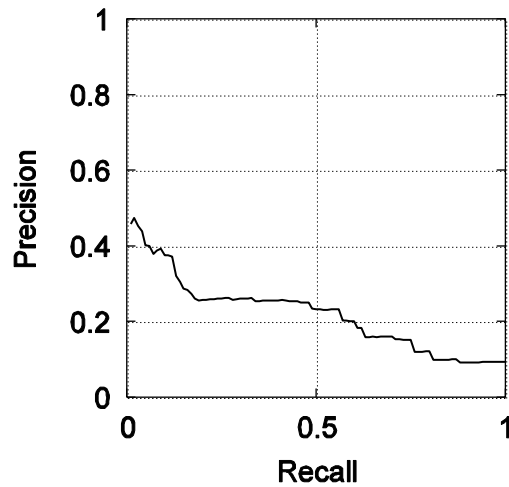


Figure 41: Recall/Precision curve for strict quantisation with *atc* weights to all subvectors—weighting amongst the subvectors: *abstract=0.01, article title=0.01, keywords=0.01, body=8.0*

INEX 2002: 01

quantization: generalized; topics: CAS
average precision: 0.196
(empty topic results ignored)

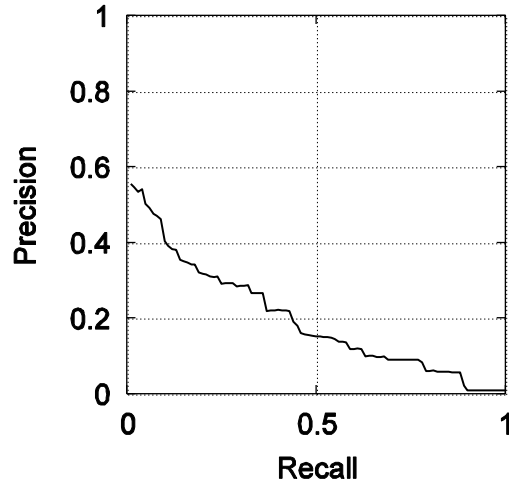


Figure 42: Recall/Precision curve for generalized quantisation with *atc* weights to all subvectors—weighting amongst the subvectors: *abstract=1.0, article title=1.0, keywords=1.0, body=8.0*

INEX 2002: 01

quantization: strict; topics: CAS
average precision: 0.229
(empty topic results ignored)

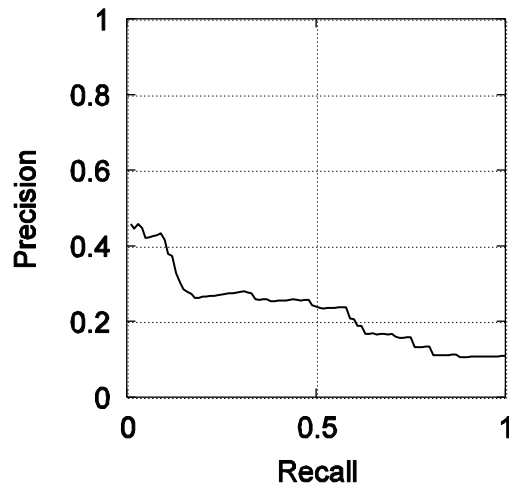


Figure 43: Recall/Precision curve for strict quantisation with *atc* weights to all subvectors—weighting amongst the subvectors: *abstract=1.0, article title=1.0, keywords=1.0, body=8.0*