Department of Computer Science
Series of Publications A
Report A-2006-3

Indexing Heterogeneous XML for Full-Text Search

Miro Lehtonen

Academic Dissertation

To be presented, with the permission of the Faculty of Science of the University of Helsinki, for public criticism in Auditorium XII of the Main Building, on 14 November 2006, at 12 pm.

University of Helsinki
Finland

ISSN 1238-8645
ISBN 952-10-3452-1 (paperback)
ISBN 952-10-3453-X (PDF)
http://ethesis.helsinki.fi/

Computing Reviews (1998) Classification: H.2.4, H.3.1, I.7.2

Helsinki University Printing House
Helsinki, November 2006 (185+3 pages)

Indexing Heterogeneous XML for Full-Text Search

Miro Lehtonen
Department of Computer Science
P.O. Box 68, FI-00014 University of Helsinki, Finland
Miro.Lehtonen@cs.Helsinki.Fi

Abstract

XML documents are becoming more and more common in various environments. In particular, enterprise-scale document management is commonly centred around XML, and desktop applications as well as online document collections are soon to follow. The growing number of XML documents increases the importance of appropriate indexing methods and search tools in keeping the information accessible. Therefore, we focus on content that is stored in XML format as we develop such indexing methods.

Because XML is used for different kinds of content ranging all the way from records of data fields to narrative full-texts, the methods for Information Retrieval are facing a new challenge in identifying which content is subject to data queries and which should be indexed for full-text search. In response to this challenge, we analyse the relation of character content and XML tags in XML documents in order to separate the full-text from data. As a result, we are able to both reduce the size of the index by 5-6% and improve the retrieval precision as we select the XML fragments to be indexed.

Besides being challenging, XML comes with many unexplored opportunities which are not paid much attention in the literature. For example, authors often tag the content they want to emphasise by using a typeface that stands out. The tagged content constitutes phrases that are descriptive of the content and useful for full-text search. They are simple to detect in XML documents, but also possible to confuse with other inline-level text. Nonetheless, the search results seem to improve when the detected phrases are given additional weight in the index. Similar improvements are reported when related content is associated with the indexed full-text including titles, captions, and references.

Experimental results show that for certain types of document collections, at least, the proposed methods help us find the relevant answers. Even when we know nothing about the document structure but the XML syntax, we are able to take advantage of the XML structure when the content is indexed for full-text search.

Computing Reviews (1998) Categories and Subject Descriptors:

H.2.4 **Database management**: Systems—*Textual databases*

H.3.1 **Information Storage and Retrieval**: Content Analysis and Indexing—*indexing methods*

I.7.2 **Document and Text Processing**: Document Preparation—*index generation, markup languages*

General Terms: Algorithms, Experimentation, Theory

Additional Key Words and Phrases: XML, Full-text search, XML information retrieval

Acknowledgements

This thesis has been a long-term project which is now finished after over three years of writing. For the professional support, I would like to thank my supervisor, Helena Ahonen-Myka, as well as the reviewers, Jaana Kekäläinen and Pekka Kilpeläinen. Additional help with the evaluation of my research was provided by Antoine Doucet and Benjamin Piwowarski who I am also grateful to.

For inspiration and Tuesday afternoon coffee, I would like to thank the do-re-mi research group: Helena, Greger, Roman, Antoine, and whoever made appearances. Special thanks to Juha and Oskari for the seemingly useless — but so frequent — debates which always took my mind away from writing and gave me new ideas and inspiration.

I also thank my fellow musicians, friends, and graduate students of HOS Big Band for their company at the rehearsals, in pubs, on concert trips, and anywhere else we went together. All that time we have spent together keeps my mind fresh and prevents me from becoming a complete nerd. I appreciate all my other friends, as well, who have not forgotten me despite my busy schedule.

Finally, I want to thank my family for bearing with me, and, in particular, my son Elliot for understanding why daddy has been so busy with work. I am also grateful for Elliot's grandparents for babysitting, and my brother for catsitting, because that has given me more chances to work on this thesis, and Diana, for help with proofreading. Whatever stress was caused by the writing process was taken care of by my feline companions — Istvan and Aziz — who are most reliable stress relievers.

Contents

CHAPTER 1

Introduction

Information Retrieval is one of the oldest research areas currently classified as a Computer Science. Despite its shorter history, the *Extensible Markup Language (XML)* — the other essential research subject of this thesis — has become widely accepted as a common format for both documents and data. Combining the two into XML Information Retrieval has given spark to publications since the turn of the millennium, which means that this active field of research is still relatively young.

The concept of a *document* makes XML retrieval fundamentally different from the traditional kind of Information Retrieval. While documents have traditionally been atomic units that are indexed and retrieved independently of each other, *XML documents* merely represent the largest logical units of XML that may comprise anything between a single XML element and a whole document collection. Therefore, we retrieve XML elements, or, more generally, *XML fragments*, instead of retrieving whole documents. Despite the structure of the retrieved answers, the format is considered irrelevant as long as the information need is satisfied. For example, relevant answers may include the chapters of a book, sections of an article, subsections of a thesis, or even a single paragraph about the topic of the query. What satisfies the information need is not the structure but the sole content.

Although the structured nature of XML facilitates queries on the document structure in addition to allowing queries on the content, in this thesis we will concentrate on the traditional *full-text* queries that only consist of search terms such as keywords and key

phrases. The notion of full-text will be used in the same sense as defined in the ISO/IEC 13249 standard [ISO00] as well as the W3C specification of the full-text extension to the XML query language XQuery 1.0 [W3C05b]. The content that is subject to full-text querying is composed of complete sentences rather than individual words. In contrast, XML documents without full-text content typically consist of data items that are rarely sentence-level objects, if they contain any text at all.

Specialising in retrieval methods for heterogeneous XML documents, which requires that the XML retrieval methods generalise, has turned out to be a challenging research direction with few successful efforts so far at tackling the problems. By definition, a heterogeneous collection of XML documents contains documents that represent several different document types. Independence of the document types thus increases the suitability of the methodology for heterogeneous XML as it can be applied to arbitrary XML documents. Moreover, the generalising methods are necessary if the document type and the document structure are unknown either to the user inputting queries or to the application evaluating them.

When we compare traditional full-text retrieval systems to those that specialise in XML, the major differences lie in the indexing methods. As a document is simply not the best or the only indexed unit of XML, we need more sophisticated ways to store XML documents in a full-text index. Keeping in mind that there are loads of previous work on Information Retrieval, we may choose from two different approaches to indexing XML documents: 1) either the legacy methodology is made aware of the XML markup, or 2) the XML documents are adapted to full-text search by reducing the significance of the markup. By choosing the latter approach, we effectively avoid the need for new methods for relevance computation, and instead, we may focus on supporting the state-of-the-art methods. For example, in terms of similarity-based relevance computation, we rely on such traditional concepts as the Vector Space Model [SWY75], which, in this thesis, is first challenged with XML documents, analysed for requirements, and, finally, adopted for XML fragments.

The main contributions of this thesis include an algorithm for dividing arbitrary XML documents into indexed fragments that contain full-text as well as three techniques of *fragment expansion*

which modify the weights of the content of the indexed fragments in order to improve retrieval precision. A great deal of the work is based on the contrast between data and full-text which plays an important role in the definitions for different types of XML fragments. By studying the properties of data and full-text content, we avoid dependence on particular document types and, consequently, develop methods suitable for heterogeneous XML documents. The main contributions will also undergo initial testing that should predict future research prospects for this topic.

This thesis is organised as follows. The research conducted for this thesis is put into perspective in Chapter 2 by introducing related work and separating the in-scope research questions from those that fall outside the scope. In Chapter 3, we review numerous definitions for XML fragments and also present a few of our own in order to have a normative basis for the fragment selection algorithm presented in Chapter 4. Once the indexed fragments have been selected, we study various ways to expand the fragment content in Chapter 5. The first contribution currently being tested is the algorithm for fragment selection, which is applied to a rather large document collection in Chapter 6. The methodology for testing how choices made in fragment selection and fragment expansion affect the retrieval quality is described in Chapter 7, which is followed by the application of the test methodology in Chapter 8, in which the results are also analysed. The final conclusions are drawn from both the results and the research in this thesis in Chapter 9.

CHAPTER 2

Indexing heterogeneous XML

Indexing text documents is found at the core of many applications of Information Retrieval, such as web search engines, web crawlers, and systems for XML Retrieval. The former have traditionally specialised in certain document formats such as HTML, PDF etc. by analysing and utilising the structure of the documents in addition to processing the plain text. The research in this thesis focuses on the latter as they index XML documents. Why we want to specialise in XML documents is motivated in Section 2.1 and the scope of this thesis is further defined in Section 2.2. In order to improve the accessibility of this thesis, important terminology and concepts are defined in Section 2.3. The common approaches to indexing and interpreting XML documents are introduced in Section 2.4 which is followed by the principles in Section 2.5 that guide the research in this thesis. Section 2.6 is a brief overview on the related work.

2.1 Motivation

The completion of the Recommendation for the Extensible Markup Language [W3C98] in 1998 was one of the first milestones in the history of the World Wide Web Consortium[1] (W3C) that started the rapid growth of the family of standards related to XML. Another significant milestone was reached in 2000 when the development of

[1] http://www.w3.org/

the SGML-based web document language HTML was superseded by that of the first XML compliant version called XHTML [W3C02] which has also been the basis for the future versions. Despite the widespread approval of the W3C specifications, the success of XML on the web has had a slow start because of some XHTML-related problems which are nicely pointed out in the famous article by Ian Hickson[2]. However, XML and the related standards have become a permanent addition to the set of common web technologies, and the status of XML as a common document format is expected to get stronger in the future. A piece of supporting evidence for this prediction is found in the syndicated newsfeeds on web pages which have become one of the first real breakthroughs for XML on the web [Rai05].

XML has had more success stories as the document format of enterprise-scale document collections such as the Vaisala[3] technical documents [LPHL02] which have been produced in XML since the early 00's. Other common examples include legal documents[4] and news articles [WDM+02]. These document collections typically cover a single domain but several different document types are possible. The XML format of the source documents gives easy access to publishing through multiple output channels in multiple output formats. In a common scenario, legacy document collections are converted into XML to be used in a system for *XML-based document assembly* [Leh01] or some other corresponding publishing system. The next step in the development of such systems is to implement the search features that make the contents of the documents accessible throughout various applications and through a single search interface.

In the near future, we are likely to see even more XML documents created by desktop applications when the *Microsoft Office Open XML Formats* become the default document formats for the new versions of Microsoft Office Word, Excel® and PowerPoint® included in the upcoming Office 12 suite. The expected time of release is scheduled for the second half of the year 2006. The wide popularity of the software makes this a significant change towards

[2] http://www.hixie.ch/advocacy/xhtml
[3] http://www.vaisala.com/
[4] http://www.legalxml.org/

more interoperable document formats. The killer application could be a high-quality search system that indexes the full-text of the locally stored documents and, in addition, specialises in the XML format.

Now that we have some faith in XML being or becoming the de facto document format in various domains, we move on to motivate XML-specific Information Retrieval. The importance of this research as well as the widely spread interest in the topic are well expressed in the great number of participating institutions in the Heterogeneous Collections Track of the INEX 2004[5] and 2005 initiatives[6] which have been the common ground for researchers in the field of XML retrieval. For full-text retrieval, the major goal of the "het track" is the development of methods that are independent of the document type [SR05]. Despite the popularity of the track, only one session in the INEX 2004 workshop was devoted to heterogeneous collections, which reflects the challenges of the field. In 2005, the schedule of the het track was stretched past the workshop so that heterogeneous collections were not discussed in a separate session despite being the topic of the most popular optional track.

The previous work in the field of indexing XML documents for full-text search can be divided into two categories: 1) traditional indexing methods for plain text and hypertext documents, and 2) quite recently developed methods for storing text in XML databases. Methodology in the first category is applicable to heterogeneous XML documents but it often falls short in its negligence of the additional markup. At best, the methods assume the HTML structure of the indexed documents. From that point of view, there is a need for the traditional methods to take advantage of the nature of XML: all the content has been marked up, and we may know more about the content than how it should be presented in a visual form. The second category consists of methods that rely on well-defined XML structures. Making these methods compatible with arbitrary XML documents is a major challenge as the efficient indexing of XML data requires highly optimised processes, which in turn shows in the dependence on a single document type. What textual databases seem to need the most is more flexibility so

[5]http://inex.is.informatik.uni-duisburg.de:2004/
[6]http://inex.is.informatik.uni-duisburg.de/2005/

that the indexing methods scale to a number of different document types without having to define the correspondence of different XML vocabularies.

The current work on XML Information Retrieval started at a higher volume in 2002 when the first round of the INEX Initiatives was organised. Right from the beginning, the participants were classified by their approaches as members of either the database community or the IR community [FGKL02]. The departure from the traditions has been slow but the communities have at least found each other: many systems combine features from both databases and traditional IR. For example, the Hyper-media Retrieval Engine for XML (HyREX) [FGG02] implements the XIRQL query language [FG01] where prespecified types of elements are indexed into data structures that correspond to tables of a database — each one for a single element type. The relevance of the elements is computed with similarity measures adopted from traditional IR as if the XML elements were independent documents. The shortcomings of HyREX are similar to those in previous research: only a single document type is supported, whereas the analysis of the XML structure is limited to defining the boundaries between the indexed document fragments which, in practice, are subtrees of the parsed XML documents.

The research presented in this thesis does not originate in the DB community, nor is it greatly inspired by research in the IR community. Instead, the original approach to XML Retrieval in this thesis comes from the XML community, which, among the INEX participants, has been a unique background once filed under the heading "Other models" in the workshop proceedings [FLM04]. Consequently, the problems that non-native XML methodology has to face can be effectively avoided by taking XML as the focal point of the research.

The role of the physical structure of the indexed documents is an example of the fundamental differences between related work and the work presented in this thesis. As XML defines the logical structure of documents, we completely ignore how the documents are stored in the file system. In the XML-centric approach, we process and analyse the parsed XML documents, whereas in related work, the indexed document collection is often described as a set of XML articles or even XML files [TO04]. Discarding the physical docu-

ment structure is essential when the methods should be applicable to heterogeneous XML documents. However, such methods have not yet been described in related work.

2.2 Scope

The scope of the research in this thesis has two dimensions. The first one in Section 2.2.1 deals with what makes XML documents relevant to this research, and the second dimension in Section 2.2.2 focuses on how the relevant XML is processed. We also identify some interesting research questions that fall outside the scope of this thesis, and as for the moment, the questions remain open for future research.

2.2.1 Right kind of XML

Heterogeneity of the XML documents was one of the most important inspirations for starting this research, so there should not be any kind of XML that our methodology could not handle. The well-formedness of the documents is a minimum requirement, though, and in this thesis, XML shall be short for well-formed XML. Validity of the XML documents is not required.

Although we want to be able to handle any kind of well-formed XML, it would be naïve to think that any kind of XML could be useful for the purposes of Information Retrieval. Therefore, we have to define the kind of XML documents that are potentially interesting to us. One way to decide between the right and wrong kind of XML is to ask whether the document is mainly intended for publication in a readable format, as that kind of documents most likely contain full-text that can be searched. Typical examples of such XML documents include books, scientific articles, magazines, journals, blogs, newsfeed articles, and other tagged text documents. Another point of view is to think of whether traditional search engines can index the document if the format is first changed into plain text. In fact, we are interested in indexing the same kind of full-text documents that were indexed before the XML era. As a summary, the scope of this research covers full-text XML documents

which are defined as any XML documents with at least some full-text content with no restrictions on the topical domains covered.

Although web pages using XML fall inside the scope, neither XHTML nor any other single document type is given special attention. Moreover, hyperlinks and other references between different documents are ignored, and documents containing shared fragments are treated as independent documents, not as a set of web pages sharing content.

The right kind of XML comes in a number of different logical formats with varying degrees of heterogeneity. At least, the following cases are possible:

A large document collection For example, a WWW-scale index contains billions of full-text documents representing several different document types.

Heterogeneous document collections Enterprise-scale document collections typically contain documents of several different document types as they may consist of several homogeneous collections.

Homogeneous document collections Document collections with a single domain usually contain documents of a single document type, but when it comes to the size, the documents still seem heterogeneous.

Oversized XML documents Sometimes it may be handy to present a whole collection of documents under a single root element, which leads to documents that are too big or too incoherent to be seen as one unit of interest. These documents have to be divided into fragments of digestible size.

Whether small XML documents with little amounts of full-text are the right kind of XML may be disputed, but including them in the scope should be rather safe as traditional methods for IR already can handle small documents, e.g., by normalising the document length. Although the documents may contain unparsed entities, such as images, video, or multimedia, we will only take into account the content as far as it can be parsed. The physical structure into which the XML documents have been serialised is irrelevant. For

example, we are not interested in how many files or entities an XML document consists of.

Equally important to defining what kind of documents are relevant to this research is to present examples of XML documents that fall outside the scope of this thesis. The first major category of XML documents that we will not be indexing is *XML data* which could be seen as the content of XML databases. Processing and querying XML data with techniques that originate in the IR communities is hardly meaningful. Besides, appropriate techniques for handling such data already exist, and, to a large extent, they can be adopted from the world of databases. Therefore, data in XML databases and XML documents will be neglected despite of its XML format. Program-readable and purely transactional XML is another major category that falls outside the scope. The typical documents in this category are often generated automatically, and transformed, interpreted, and processed by applications. The common document types for program-read XML include XML Stylesheets[7], SOAP[8], and SMIL[9].

One of the few classifications for the document types of XML was presented by Choi who categorised DTDs into three classes according to their purpose [Cho02]:

1. Database-like data, e.g. baseball statistics.
2. Data interchange between applications, e.g. bookmark exchange.
3. Document markup and metadata, e.g. dramatical plays.

Although his examples may seem outdated, the categorisation is still sound as it differentiates data from text documents. As we already discarded the first two categories as irrelevant, the third category is the only one of interest in this thesis. However, the problem with these categories is the same problem as with the scope definition: not all XML documents belong to any single class. For example, Really Simple Syndication (RSS)[10] documents which

[7] http://www.w3.org/Style/
[8] http://www.w3.org/2000/xp/Group/
[9] http://www.w3.org/AudioVideo/
[10] http://web.resource.org/rss/1.0/spec

describe syndicated XML feeds are processed by applications, but they also contain both data and full-text. With the support of XML Namespaces, we may have arbitrary combinations of different kinds of content in one XML document.

The grey area of the *document type spectrum* a.k.a. the *XML usage spectrum* has been recently described in the literature as a continuum instead of a binary division into data and documents. In 2003, Charles Goldfarb stated that "*...there is no longer a difference in kind between the two, only a difference in degree*" [Gol03]. Further support was provided by Glushko and McGrath in 2005 who find that "*...difficult distinctions arise in the middle of the Document Type Spectrum where documents contain both narrative and transactional features*" [GM05]. The documents that belong to this grey area are especially relevant to the thesis due to the challenge of separating the document fragments that can be indexed for full-text search from those that could be queried as data.

2.2.2 Adapting XML to full-text search

The right kind of XML is rarely ready to be indexed for full-text search, as it is. For one thing, it is usually too heterogeneous, and for another, it has unused potential that goes beyond the capabilities of traditional indexing methods for plain text or even hypertext documents. In this thesis, we investigate two major issues concerning full-text documents in XML format. The first research question concerns **how the right kind of XML can be automatically recognised**, and whether the recognition of the wrong kind of XML also is necessary. The second research question is about **how the heterogeneous XML can be indexed so that the traditional indexing methods are applicable**, in particular those that regard documents as a bag of words. In the approach chosen for this research, XML documents are indexed as a collection of independent XML fragments which have also been called *index nodes* in recent literature[11] [FG01].

Selecting the indexed fragments is one of the key areas of interest, as the concept of a document has significantly changed from that in

[11]The term "index node" was originally introduced by Fuhr and Großjohann in the context of the query language XIRQL.

traditional Information Retrieval. Web search engines have faced a similar problem which has led to the block-level searching of web pages [CYWM04]. Another feature that we want to adopt from the modern web search methodology is that the structure of web pages and HTML documents is analysed to the fullest by www-robots and search engines that index web pages. A similar yet different analysis of XML documents helps apply different weighting schemes to the indexed XML fragments.

Combining the traditional indexing methods with the analysis of the XML structure is not a trivial workout, either, as it involves the removal of the XML markup without losing the information it encodes. Moreover, the resulting plain text fragments should have the properties of traditional documents so that they can be prepared for querying. Whatever can be done to XML documents before the markup is removed is included in the scope of this thesis.

The following research questions are related to the major topics of this thesis either through the settings of the evaluation or through the challenges in realistic application areas.

Element score computation Various language and retrieval models as well as techniques related to Natural Language Processing (NLP) may be applied to the computation of element relevance. However, investigating or evaluating such methods is a wide area of research that, though closely related, is not regarded in this thesis in much detail.

Element score combination and propagation Related to the computation of the element scores, we need score combination or propagation methods when the relevance to a query is computed for disjoint fragments only. The relevance of the parent elements is computed recursively with these methods based on the relevance scores of the children. Upward propagation of scores is also known as the *augmentation* method for which various weighting techniques have been proposed [FG01, GS02, Leh05, TSW05]. In addition, Mihailović et al. considered the downwards propagation of scores [MBHA05]. When all the elements in the XML documents are associated with a relevance score, it is rather simple to determine the relative ranking for elements representing different granularities.

Presentation of the search results Depending on the user model, we may request the search results to be presented as a list of pointers or as a list of links to relevant standalone documents. The biggest challenge lies in the results that originate in the same document or even overlap with each other. Grouping these results by the source documents easily interferes with the original relevance-based ranking.

Formatting XML without stylesheets The standalone documents consist of arbitrary XML fragments. Even if the document type is known, stylesheets may not be available or they may not be applicable to a single document fragment, which makes other methods necessary for displaying the search results. Ahonen et al. proposed an approach as early as in 2000 [AMHHK00] but few other proposals have been seen since then.

Anti-spamming techniques As indexing methods develop, so do the spamming techniques which are supposed to increase the visibility of the documents by deception. The anti-spamming techniques are often one step behind in the race. Nevertheless, more widespread use of XML search engines is required for the race to begin.

Although some of these topics have already been covered in recent literature, all of them are still subject to further research.

2.3 Terms and definitions

This section is devoted to descriptions and definitions of the essential terminology used in this thesis. Most of the terms introduced are related to XML which is not always clearly described in the scientific literature. The use of the following terms and concepts in this thesis should not contradict with how the identical terms are used in the W3C specifications, however, there might be examples of conflicting usage of the terms in the scientific literature.

XML document Access to document content including XML elements and text is provided through XML documents which

are the largest units of XML that have to be processed at one time. In this thesis, bigger units such as XML collections are processed serially, one XML document at a time.

XML element XML elements consist of the start tag, element content, and the end tag, with the exception of empty elements that only consist of the empty element tag.

Content model The content model of an XML element defines what kind of content is allowed for a particular element type, such as the names and order of child elements, text content, or empty content.

XML tag A tag name enclosed in angle brackets makes an XML tag. In addition, end tags come with a slash before the tag name and empty element tags with one after the tag name.

XML fragment The somewhat vaguely defined unit which can be bigger than an XML element but smaller than a whole XML document is called an XML fragment. Numerous definitions are regarded in Chapter 3.

Root element The outermost XML element such as the first element of an XML document is the root element of the document. XML fragments may have more than one root element, but the definition is similar: elements not enclosed within other elements are fragment root elements.

Text The text content of an XML document may occur in two kinds of environments: in the element content and in the start tag as an attribute value. Any other content including comments and tag names is not considered to be text content.

Nodes XML documents are often discussed in their parsed tree representation, which is when documents, elements, and text all become nodes of the document tree.

Node type Each node is associated with a type, e.g., elements are parsed into element nodes, and text is parsed into text nodes. The capitalised Node Type refers to the DOM Node Type such as the Element Node and the Text Node.

Overlapping elements The content of nested elements comes with overlap by definition. When document trees are considered, ancestor nodes overlap with their descendants.

Inline elements An element with text node siblings is an inline element.

Block-level elements Any element with text node children but without text node siblings can be called a block-level element. In practice, a block-level element starts a new line.

2.4 Specialised vs. generalised methods

What we know and what we can safely assume about the indexed XML documents set the basis for the methods chosen for the index construction. As a rule of a thumb, the more we know about the document structure, the more we can specialise in the type of the indexed documents, and also, the more we assume about the structure, the more we actually do specialise. In this section, we study what specialising in a document type involves. First, we regard the problem from the viewpoint of a single document type in Section 2.4.1. Several different document types are considered in Section 2.4.2 which is followed by discussion about generalised methods in Section 2.4.3, where no specialisation is allowed at all.

2.4.1 Specialising in a document type

Indexing methods that specialise in a certain document type are appropriate when the document type is known or at least assumed. For example, a famous web search engine that specialises in the html document type has made the following announcement:

"*Google*[12] *goes far beyond the number of times a term appears on a page and examines all aspects of the page's content (and the content of the pages linking to it) to determine if it's a good match for your query.*"

When interpreted, the message says that besides relying on methods for flat text retrieval, Google analyses the structure of the

[12] http://www.google.com/

HTML (and XHTML) documents including assumptions about the semantics of the structure.

Indexing methods relying on information specific to a document type require a manual analysis of the DTD or the Schema definition. Valid document instances should also be analysed if very irregular structures are typical of the documents. Based on the analysis, element types are given meanings, roles, and heuristic indexing instructions. These guidelines for indexing could declare, for example, that the `<title>` element shall contain the title of the document and it shall be given an appropriate weight in the index, or that `<p>` element shall contain full-text to be indexed and only the element content shall be considered [LZC04].

Applying the specialised indexing methods requires the identification of the document type, which is usually based on the document type declaration[13] that refers to a DTD, or a reference to a Schema definition. If the DOCTYPE declaration is missing or if the document is not valid, we need other ways to identify the document type of a single document. The name of the document element and other common elements usually provide useful evidence when determining the document type. Alternatively, we may assume the document type, in particular when validity is not required. For example, web search engines and browsers typically assume that files with the extension .html are HTML documents. Internet Explorer (IE) 6.0 goes even further by ignoring both the extension .xml and the XML declaration, given that the root element is html, and the MIME type `text/html` is assumed for all XHTML documents. Applications can specialise in any XML document type in a similar fashion.

2.4.2 Mapping multiple document types

When the indexed documents represent several document types, we may want to specialise in each type separately. If the types are similar enough, we can save some of the trouble by mapping equivalent or compatible elements (or paths) between different document types and translate either the queries or the resulting answers accordingly.

[13] In XML documents, the declaration starts with `<!DOCTYPE`, and it often follows the XML declaration.

The mapping is performed manually, or semi-automatically at best, after a careful analysis of the document type definitions. Specialising in each document type thus requires a learning process which can hardly be automatic or unsupervised. Therefore, specialised methods are not suitable for anything larger than enterprise-scale collections with a limited number of different document types and low volatility in document type definitions.

The actual mappings include the typical one-to-one, one-to-many, and many-to-one relations between path expressions that consist of elements, attributes, and attribute values. The problems and challenges are also typical as they are similar to those in federated databases [SL90, Col97], ontology mapping [Cha00, DMD+03, MFRW00], and view generation [ACM+02, JH01]. Thanks to the mappings, both federated databases and heterogeneous XML collections are queried and searched using a common query language with the illusion that the user is accessing either a single database or documents of a single document type. This way, a heterogeneous collection of XML documents is treated as multiple homogeneous collections where document-type specific assumptions are applied to each homogeneous subcollection.

How practical and useful the mappings turn out to be depends largely on how accurately the document type definition describes the document instances. The success of a mapping can be estimated by measuring the amount of variance allowed in valid documents. If the documents show great structural variance, the DTD does not accurately describe all the valid documents, and the mappings cannot thus be very reliable. However, a bigger challenge is caused by document types that are so incompatible that mappings between types cannot be defined. For example, mappings between document types that describe the presentation of the content and those that describe the semantics thereof are nearly impossible to define because of the different semantics of the XML vocabularies.

Mappings can be defined within a single document type, too. According to the INEX 2003 guidelines for topic development [KLM03], the interpretation of the Vague Content-And-Structure (VCAS) queries implies the equivalence of elements that belong to the same group. For example the section elements `sec`, `ss1`, `ss2`, and `ss3` are considered equal in the evaluation of the search results. The equivalence classes of the INEX VCAS queries were defined by a

team of INEX activists who analysed the groupings of the element type definitions in the DTD. If equivalence classes are defined for other document types, the similarity of the content models may also be taken into account.

2.4.3 Generalising to arbitrary document types

Even if the indexing methods could be trained to handle documents of several different types, we need *generalised methods* that make no assumptions specific to a document type. In particular, XML documents with full-text content often employ document types that allow irregular and unpredictable structures to such an extent that the assumptions about the document type go simply wrong. For example, a typical element type definition does not set any limit for the size of its full-text content. Although the size of an article element in the INEX test collection[14] varies between 186 and 228,112 characters, it is a common misconception, e.g. in [FGKL02], that there is a one-to-one correspondence between scientific journal articles and XML elements called `article`. By looking more closely, one finds a diversity of papers between the `article` tags including indices, errata, lists of reviewers, and calls for papers.

Besides the intentional heterogeneity of document structures, there is a lot of room for unintended inconsistency due to the different practices of document authors. In particular, automatic conversion processes from other document formats into XML further increase the risk of inconsistent usage of the designated markup. Again, the INEX test documents are a good source of examples. The DTD defines certain element types for titles, but not all of the title elements contain titles in the actual documents. Moreover, some of the titles in the documents are not contained in any title element. Such findings make the reliance on document type definitions seem quite prone to errors. Yet another reason to avoid the specialisation into certain document types is spamming which can be based on the misuse of element names. The HTML `meta` (keywords) element used to be a good example of spamming that is specific to a document type.

[14] See Section 6.1 for more details.

Even when we do not specialise in any particular document type, we do specialise in XML documents with full-text content. Not much can be assumed about the vocabulary of the XML structures, but we do have a lot to analyse in the pure XML syntax of the documents. For example, every XML element has a content model, a size, and a position in the document which is related to the other elements of the document. The elements may also have attributes and text content. Both elements and attributes may be assigned *types* that are common to all XML documents, e.g. the atomic data types of XML Schema. Assumptions about the *meaning* of content models, size, types, etc., concern all XML documents regardless of their document type, and they are thus applicable to heterogeneous XML documents. The major difference from the specialised methods is that the assumptions only apply to the document instances, whereas the document type definitions need not be analysed at all.

Not assuming anything about the document type serves the purpose of developing indexing methods that apply to arbitrary XML documents, e.g. all the documents in a heterogeneous collection. However, regularities and heuristic rules that are based on analysing the documents might not be very reliable if too much heterogeneity is involved. For example, the size and the type of the text content is strongly dependent on the language, text orientation, the character set, and the encoding. If all the details were taken into account, the complexity of the methods would increase to such proportions that the specialised methods might even be less error-prone in comparison. A possible solution to the complexity of the generalised methods could be the combination of specialised and generalised methods. For example, if the DTD is available, we may scan the DTD for attribute list definitions and learn which ones are of types ID or IDREF(S), which is faster and more reliable than trying to detect the same attributes in the valid XML documents [BM03].

These hybrid indexing methods that allow the limited use of the document type definitions are appropriate in cases that involve a limited number of document types, e.g. an enterprise-scale document collection. What is a reasonable amount of specialisation depends on the nature of the collection: how regular the document structures are, how the XML is produced, etc. If the documents are still too heterogeneous for the indexing methods, we may have to resort to indexing only what can be reliably indexed and recognising

the rest of the documents where the methods do not apply.

2.5 Design principles

So far, we have described what kind of XML we want to process and what kind of processing awaits the indexed XML documents. In the previous section, the common approaches to indexing XML were compared with each other. In this section, we make the necessary choices and set the goals for the research conducted for this thesis. The major pinciples are introduced in the following paragraphs.

Information specific to a document type is disregarded. Independence of document types and document structures is considered important because we want to be able to index heterogeneous XML collections. By applying generalised methods to the documents actually makes them seem homogeneous. Meanwhile, analysing document types manually becomes unnecessary.

Documents are indexed independently of each other. The order of indexing should make no difference in the outcome, which implies that we can index one XML document at a time and that the document representation in the index is independent of the other indexed documents. Consequently, the indexing methods are scalable, and the large size of a document collection will not be a problem.

Information accessible through the W3C DOM representation only is considered. Because validating XML parsers do not have to report details related to the physical properties of the document such as ignorable whitespace, parsed entities, and file names, we cannot rely on information about the physical structure of the documents. However, we do assume that the logical structure of the documents is provided by a compliant DOM parser.

Traditional relevance and document models are applied. Despite the differences between highly structured XML documents and the traditional plain text documents, existing methods for Information Retrieval should be applicable after some adaptation of the XML documents. The adaptation of XML documents to the models of traditional IR is actually one of the main contributions of this thesis.

The indexing methods are independent of queries. Some users require more in their query than just the topical relevance of the answers: they also specify the size of granularity of the relevant answers. While some systems have different indices for different tasks, e.g. finding paragraph-sized or section-sized answers, our goal is a single index that is used with all the queries. The size of the returned answers is determined by the relevance of the answer instead of by the constraints in the query.

Other important principles might also be recognised, but those that were mentioned are those that together make this research different from the related approaches. Some principles that have been considered important elsewhere have been intentionally left out. For example, no particular query language is given any special support, and in addition, the structural constraints that may occur in full-text queries are not taken into consideration. As these goals were set before the actual research was started, they show direction to this research more than dictate it.

2.6 Related work

XML Information Retrieval has gained popularity as a field of research since the SIGIR workshop on XML and Information Retrieval in 2000 [BYFSDW00, CMS00], followed by sequels in 2002 [BYFM02a, BYFM02b] and 2004 [BYM04]. The number of implementations of experimental XML search engines was boosted by a common ground for the evaluation provided by the INEX initiatives in 2002–2005 [FGKL02, FLM04, FLMS05, FLMK06] which further inspired the INEX workshop on Element Retrieval Methodology in Glasgow in 2005 [TLF05]. These efforts have resulted in a great number of publications in the field of full-text search of XML documents. After reviewing the most important work in the field in Section 2.6.1, we proceed to the related research areas of structured XML retrieval and XML mining which are reviewed in Sections 2.6.2 and 2.6.3.

2.6.1 Unstructured search

Full-text search of structured documents has been studied since the early 90's [Wil94]. Countless reports and articles have been written since then about searching HTML documents, and more recently, also about keyword queries on XML documents. Because of the explicit structure of XML documents, it has been common to build keyword indices either on a fixed set of element types [DALP04, FGG02, MM04], a configurable set of element types [LZC04], or an unrestricted set of element types [HLR04, WMSB04]. Indices that are built on more than one element type often include the same content multiple times because of the nested structure of elements. In order to address the issue of overlapping content in the index, systems like EXTIRP specialise in defining disjoint index units [Leh05]. More details will be provided in Chapters 4 and 6.

Regarding the content of a full-text index, Kamps et al. presented some intuitively useful results after studying length normalisation in XML retrieval [KdRS04]. One of their interesting observations was that units shorter than 50 or even 60 index terms can be left outside the full-text index without the cost of losing potentially relevant answers. Similar cut-off values will be studied in this thesis, though the lower bound is measured in characters rather than in index terms.

The important role of structure when indexing and searching XML has been acknowledged by several research groups. Typically, the content of some elements is given more weight than that of other elements. The heavier weights have been associated either with certain element types [LZC04, MJKZ98] or with the relevance of the surrounding content [AJK05]. Term selection has also gained some new flavour from XML. The same term in a different context is successfully considered a different term [CMM$^+$03, LZC04, WMSB04].

Quite separated from the researcher communities of INEX and SIGIR, several vendors have developed and incorporated search features into their enterprise-scale document management systems. The commercial systems that support the retrieval of XML elements instead of whole documents include the MarkLogic Server[15],

[15] http://www.marklogic.com/

TEXTML Server by IXIASOFT[16], and IBM WebSphere® Information Integrator Omnifind™ Edition. Of the public search engines that specialise in XML, at least Feedster[17] and Feedplex[18] are worth mentioning. Both of them are tuned for indexing and searching XML feeds.

2.6.2 Structured XML retrieval

One of the advantages of XML over plain text document formats is the structure which can be tested in the queries assuming that some structural constraints are set in addition to the conventional keywords specifying the information need. While unstructured queries can be expressed in keywords and keyphrases and possibly some logical operators, the structured queries usually require a more complex query language that comes with a syntax for presenting the structural requirements both for the queried documents and for the retrieved answers. The most widely used XML query language is XQuery [W3C05a] which has been extended to provide the full-text search capabilities in work-in-progress such as the W3C XQuery 1.0 and XPath 2.0 Full-Text [W3C05b], TeXQuery [AYBS04], and FleXPath [AYLP04]. Earlier proposals for an XML query language include Lorel [AQM+97], ELIXIR, an Expressive and Efficient Language for XML Information Retrieval [CK01], which extends the query language XML-QL [DFF+99] with a textual similarity operator, Quilt [CRF00] which is one of the predecessors of XQuery, and XIRQL [FG01] which supports features specific to Information Retrieval such as weighting and ranking. One of the most recent proposals is XSQuirrel [SA05] which specialises in sub-document queries.

A common approach to processing queries that contain path expressions is to index the data and answer the queries by probing the index only. A path index is supposed to speed up the evaluation of the path expressions, but it also reduces the size of the index considerably in comparison with a flat document index. Simple path queries without branches are supported by one of

[16] http://www.ixiasoft.com/
[17] http://www.feedster.com/
[18] http://www.fybersearch.com/feeds/

the earliest path-oriented document summarisation structure called *DataGuides* [GW97] which indexes each distinct path in the XML document. DataGuides has been further developed in several occasions [WMSB04, EOY05, WSM05]. For example, the Index Fabric [CSF+01] indexes frequent query patterns that may contain wildcards in addition to indexing the paths included in the DataGuides.

Milo and Suciu proposed a structure similar to DataGuides called 1-index [MS99] which considers the *label paths* from document root to each element. Identical label paths form the equivalence classes that together compose an accurate structural summary of the XML document. The large size of the 1-index is optimised in the A(k)-index [KSBG02] which only considers label paths that are no longer than k, thus making it an *approximate index* for paths longer than k. The D(k)-index [CLO03] is an optimised version of the A(k)-index in that the index graphs of the D(k)-index are adapted according to the query load and irregularity of query patterns. The M(k)-index [HY04] further develops the D(k)-index by allowing different values of k for different nodes and even multiple values of k are possible in the M*(k)-index [HY04] that consists of a collection of M(k)-indexes. Yet another path index that is aware of the workload was named APEX [CMS02]. It enhances the path summary with a hash tree that enables a more efficient querying of frequently occurring paths.

Support for queries with branches is included in most of the more recent index structures, such as the F&B Index [KBNK02] which has been optimised in its disk-based version where also coding schemes are integrated into the index [WWL+05]. Zou et al. proposed a two-level tree structure called Ctree for indexing XML [ZLC04]. The group level of Ctree consists of the path summary of the document, whereas the element level contains links to parent and child elements. Besides paths, an index can also be built on *nodes* with various numbering schemes [LM01, ZND+01] or *sequences* [MJCW04, PK05, RM04, WPFY03].

2.6.3 XML mining

Methodology on document analysis, recognition, and understanding is traditionally applied to hardcopy documents [Cur97, TDL+94, Lub04] but, more recently, also to structured documents [HB03,

MYTR03]. When corresponding methods are applied to XML documents, the research area can be called XML mining in a similar fashion to how HTML documents are subject to web document mining. XML mining is an emerging field of research, which has lead to new arenas for presenting contributions, such as the first workshop on Knowledge Discovery from XML Documents (KDXD)[19] in 2006.

Despite the decent body of work in the field, there have been no widely accepted definitions for XML mining tasks [ZY04]. A common data mining task — data classification — and its application to XML data was studied by Zaki et al. [ZA03]. They presented an algorithm (XRules) that finds structural rules by analysing frequent subtrees in XML documents. The resulting classifier is able to predict the class of the data content by only considering the XML structure around it. While XRules seems appropriate for the data classification task, it does not account for the challenges presented by the full-text content of XML documents. Mining for association rules is another common task that has also been applied to XML documents that contain data [BCKL02, WD03]. However, research on XML Information Retrieval does not benefit much from the success of XML mining methods as long as the emphasis falls on XML data instead of XML full-text.

The common factor with XML mining and the research in this thesis is that the methods for XML mining do not assume much more about the data than the XML syntax. In a similar fashion, we only assume the XML syntax of the indexed full-text content in the rest of this thesis.

[19] http://sky.fit.qut.edu.au/~nayak/KDXD06/overview.html

CHAPTER 3

XML Fragments

It was not long after the publication of the XML Recommendation [W3C98] that the concept of an XML fragment became known across various contexts and application areas. The well-defined terms of XML document and XML element were simply too inflexible, i.e. they could not denote arbitrary portions of XML markup that covered several elements but not a whole document, so it was only safe to talk about vague XML fragments with no conventional definition. In Section 3.1, we first review the related work and see diverse definitions for XML fragments with a focus on those that are indexed, those that contain full-text, and those that can be detected. What kind of fragments are relevant to information retrieval, and what qualities are required, preferred, and favoured are then examined in Section 3.2. Measuring and defining the size of the relevant fragments is the subject of Section 3.3, finally followed by heuristics in Section 3.4 for automatically determining whether the fragment contains full-text.

3.1 Motivation for fragmentation

Various parties have found different ways to divide structured documents into fragments. Not all of the related work specifically define XML fragments, but, more general fragments of structured full-text documents are described instead. Because the definitions are often applicable to some XML documents, e.g. XHTML documents, it is

worthwhile to investigate whether they can be applied to arbitrary XML documents, as well.

The definitions for a document fragment vary according to the requirements specific to the application area. A selection of the proposed definitions is reviewed in this section, classified by the needs behind fragmentation. The common purposes of use include fragment interchange in Section 3.1.1, caching of web pages in Section 3.1.2, topical segmentation in Section 3.1.3, document adaptation to low-resolution displays in Section 3.1.4, and fine-grained search in Section 3.1.5. What the definitions have in common and how they differ are summarised in Section 3.1.6.

3.1.1 Underspecified standard

One of the early efforts in the field was put forth by the World Wide Web consortium (W3C)[1] which chartered a working group for XML fragments. The motivation for the intended specification for XML Fragment Interchange came from purely XML-oriented needs: how to view, edit, or send entities of XML documents without having to view or edit the entire document. Although the specification has not been rewarded a status higher than a Candidate Recommendation and although there has been no active work on the document since February 2001, some terminology was laudably defined. The terms that are here specified will be used and further clarified throughout this thesis.

According to the definition in this W3C work-in-progress [W3C01], a *well-balanced fragment* matches the production of element content:

```
[43] content ::= CharData? ((element | Reference |
                 CDSect | PI | Comment) CharData?)*
```

If a fragment contains any part of XML markup, it has to contain all of it. In case of an XML element, all of the start tag and the end tag must be included in a well-balanced fragment. There can be several root elements in a well-balanced fragment which can also be empty or contain text but no root element. The object representing the fragment removed from the source document is called

[1] http://www.w3.org/

the *fragment body*. Information not available in the fragment body but available in the complete source document is called *fragment context information*. For example, it includes information about content that is referred in the fragment body. The storage object for a single fragment is called the *fragment entity*. Fragment entities are the units of fragment interchange.

3.1.2 Optimisation of bandwidth

In order to avoid the overhead of frequent updates caused by websites with dynamic content, Ramaswamy et al. propose an algorithm for dividing web pages into update-friendlier cache units [RILD04]. They consider each web page of a web site a candidate fragment. The candidate fragment is detected as a cost-effective cache unit if it meets the following criteria:

- It is shared among at least two distinct fragments, which constitutes the *sharing factor*.
- It is maximal: There is no other fragment that contains the candidate fragment and is also shared among the same fragments.
- It has distinct personalisation and lifetime characteristics so that no ancestor fragment is updated at the same frequency of time.

The definition also includes a minimum fragment size as a parameter subject to optimisation. The algorithm is proven useful in the experiments as it reduces the amount of required disk space as well as the number of bytes transferred between the cache and the server. The authors assume that the cached documents are well-formed HTML documents but claim that the approach is applicable to XML documents, too.

Challenger et al. also take into account the rate at which the content of different parts of web pages is updated [CDIW05]. They categorise web page fragments into computer-generated *immediate fragments* which have a relatively short lifetime and *quality controlled fragments* which require a longer publishing process including proof-reading and revision. This binary division is interesting

as the immediate fragments are typically rich of data whereas the quality controlled fragments tend to contain human-written full-text. The recognition of the different fragment types is not automatic as it is based on *object dependence graphs* (ODGs) which are specified by the users such as web page designers.

3.1.3 Topical diversity

Web search engines have long had to deal with the problems caused by multiple-topic and varying-length web pages. Several different solutions have been proposed. One of the earliest efforts involved the HITS algorithm that categorises web pages into *hubs* and *authorities* [Kle99]. A page with a good collection of links has a high hub score whereas a popular page or an authoritative source of information has a high authority score. In order to improve the quality of the hubs, Chakrabarti's algorithm disaggregates web pages considered hubs into coherent regions by segmenting their DOM trees [Cha01, CJT01]. The segmentation results in improvements in topic distillation, and it can also be used for extracting relevant snippets from partially relevant hubs.

Web mining tasks have also inspired the development of page segmentation algorithms. The VIsion-Based Page Segmentation (VIPS) algorithm [YCWM03] operates on the semantic tree structure extracted from the web page by its visual presentation. The nodes in the semantic tree correspond to web page blocks enabling both block-level link analysis [CHWM04] and block-level web search [CYWM04]. Song et al. further develop the approach by not only considering the layout of the page, but also analysing the content of the blocks [SLWM04].

The common factor in these approaches is that they all divide topically incoherent pages into coherent fragments in order to improve the precision of Information Retrieval and also, in order to provide the readers with digestible portions of information. Whether the division, segmentation, or fragmentation, is based on textual content, page-to-page references, or page structure depends on the implementation [BYR02].

3.1.4 Displays with a limited resolution

Small displays have limitations in the amount of information that can be shown without too much user interaction such as scrolling. WebTV's that are viewed from distance have to deal with similar issues because of their low resolution. Although many approaches are based on thumbnail pictures, the unlimited size of web pages requires that the scalable methods include the fragmentation of web pages into smaller blocks.

The content of the page is analysed in some methods, such as single-subject splitting and multi-subject splitting [CMZ03], whereas the structure of the document and the page layout are considered important in others. For example, Hoi et al. present an automatic Document Segmentation and Presentation System (DSPS) [HLX03] which analyses the hierarchical document structure and, given the display size, divides the page into logical segments. DSPS specialises in HTML documents as it relies on HTML tag names in the analysis. Another example was proposed by Xiao et al. whose page splitting algorithm transforms pages into box-shaped blocks where the screen size, block size, number of blocks and the semantic coherence between the blocks are taken into account [XLHF05]. The algorithm starts from the VIPS tree representation of the web page which is split into blocks with a non-binary variant of the binary slicing tree algorithm [LW01].

3.1.5 Structural heterogeneity

The XML specification sets no limits on the size of XML documents which can contain anything from short messages to large reference books. As a consequence, the size of an indexed or retrieved document is no longer determined by the author of the content but by the retrieval system instead. Besides being a necessity, the new responsibility of the systems is seen as an opportunity, as well. When the documents are indexed as fragments, the users can be given direct access to the relevant fragments instead of making them browse through whole documents.

A typical challenge when searching scientific articles is that we may want to skip the parts that we are familiar with in addition to ignoring the parts that are plain old irrelevant. For example,

scientists tend to motivate their research with an introduction in the beginning of each publication. Though useful to the big audience, the peer researchers may want to skip the introduction and go straight down to business with the sections where the contributions are detailed. This is a realistic scenario when the size of the returned answers is decided independently for each query.

3.1.6 Common and distinctive features

The definitions in the related work are wrapped up in this section by analysing what they have in common and explaining how they are different. The first detail that all the definitions share is that the content of any single fragment comes from one XML document — the *source document* according to W3C. The source document itself is the biggest fragment of any XML document. Literally speaking, an XML fragment is a well-defined part of an XML document, however, all kinds of well-defined parts are not considered XML fragments by all the different definitions. For example, most definitions either define or at least imply a minimum size requirement for the fragments that are being defined.

The definitions differ in the criteria that divide the documents into fragments. The criteria are categorised by what is analysed or simply measured in the documents as follows:

Analysed content Approaches based on topical classification of the textual content may completely ignore everything but the plain text content in the analysis as they try to detect the topic boundaries and segment the text into topical segments.

Analysed structure If we know the document type of the structured document, we may ignore the content and only look into the document structure, identify tag names, analyse link relations, etc. As a simple example, whole journal documents are divided into article-sized fragments by separating the subtrees corresponding to the article elements recognised by their tag name.

Analysed updates Detecting the frequency of local changes in different parts of the document is useful when the contents are updated regularly. According to the assumption behind

update-based fragment detection, the updates always concern whole fragments regardless of the size of the update.

Measured qualities If not much is known about the content or the structure of the documents, we may settle with something quite simple: instead of complex analyses, we can measure simple properties of the documents and its potential fragments such as fragment size and its tree distance from other fragments. The parameters for the division criteria may be set by hardware, such as the display size, but they may also be based on statistics.

The criteria belonging to different categories may also be combined in any possible way. One of the contributions of this thesis is fragment selection algorithm which is mostly based on measured qualities but, to some extent, also on analysing the document structure.

What also makes the definitions different is the role of markup in the fragment. Markup is essential if the fragments are reused, e.g. sent to an XML-aware application, whereas displaying raw text or indexing the full-text content does not necessarily require any information about the tags and XML attributes.

3.2 Fragments for full-text retrieval

Because none of the definitions in related research quite matches our needs, a new definition that is applicable to XML retrieval is proposed. The requirements for the definition are vague at the early stages of the research, so we continue to characterise the XML fragments that are relevant in the context of this thesis in a similar fashion to how the relevant source documents were described in Section 2.2.1. The biggest difference from the definitions in related work lies in the application area, which also shows in the processing methods that are applied later on. In Section 3.2.1, we look at methods for determining whether an arbitrary XML fragment is suitable for full-text retrieval, whereas the most typical examples of such fragments are described in Section 3.2.2.

3.2.1 Complete units for Information Retrieval

We start from the W3C notion of an XML fragment and extend it into our own definition for an *XML Full-Text fragment* which is more appropriate for the needs of Information Retrieval.

Any well-balanced fragment that can without the start and end tags function as an independent unit in some context or use case can be considered an XML Full-Text fragment.

The extended definition adds two requirements to the XML fragments of the W3C: one for the independence of the text content, because the W3C fragments may contain too little text to stand alone, and another for the insignificance of the element names, because the content of the W3C fragments may have to be interpreted according to the tag names. The interpretation of a full-text fragment is independent of tag names as they often are instructions for displaying the content, whereas the content in data fragments is interpreted according to the tag names belonging to the absolute XPath expression of the element, e.g. `/article/author/lastname`. Having to be well-balanced is a syntactical requirement which is ignored when the tree representations of the document are regarded because every element node in a document tree is considered well-balanced when serialised.

Determining the independence of a fragment may seem like a matter of imagination: can we think of a context or a use case where the fragment qualifies? In order to make the definition clearer, we list three test questions which can be applied to a potential XML Full-Text fragment:

(Q1) **Is it meaningful to retrieve the fragment on its own?** Independent fragments are meaningful when returned by a search engine, but it is also meaningful to return a good starting point for navigation in the case that the fragment is closely tied to other fragments.

(Q2) **Is it meaningful to index the fragment as an independent unit?** Coherent or specific fragments can be indexed as atomic units whereas exhaustive fragments that combine diverse sources of information or fragments with several links pointing to more specific fragments cannot be considered

atomic enough because they might not allow for the dynamic determination of the size of the retrieved unit.

(Q3) **Could the fragment be regarded as a unit for fragment interchange or reuse?** Small but concise fragments are simple to reuse, which is why systems with reusable documentation fragment the documents accordingly [Pri01]. In addition, when the indexed fragments are the same as the reused fragments, duplicates are easy to detect and remove.

One positive answer to these questions is enough for the fragment to qualify as an XML Full-Text fragment, but in many cases, several positive answers are expected as the set of questions is somewhat redundant. For example, if it is meaningful to index a fragment as an independent unit (Q2), it is most likely meaningful to retrieve the fragment as such (Q1). Two different questions are useful because of their different points of view, however. Question (Q1) can be rephrased into the question "Can we find a query..." whereas Question Q2 becomes "When the whole XML document is indexed, is this fragment among the indexed ones?"

The definition of an XML Full-Text fragment leaves out fragments that are too small in that they contain too little information or that they are too closely tied to other fragments. For example, the content of a too small fragment can be ambiguous when taken out of the context which is found in the surrounding elements. The smallest fragments that satisfy the conditions are minimal complete units of full-text, but also bigger fragments qualify.

The definition of a well-balanced fragment is useful as it requires completeness from the markup, but as such, it is not sufficient. Well-balanced fragments may not contain any elements at all, and even with element content, there can be text outside the root elements of the well-balanced fragments. These orphaned text nodes do have original parent nodes (often also more sibling nodes) in the source document, which can be considered a dependence relation, and as such, the well-balanced fragments may be both incomplete and dependent on the source document. Moreover, allowing text nodes before and after the fragment root elements complicates the issue of fragment identification. The nodelist that identifies the fragment would no longer contain only element nodes which are simple to address, but it would contain both root elements and

text nodes. For these reasons, text nodes are only welcome as a descendant of a root element of an XML Full-Text fragment.

More clues about which fragments are suitable can be found in the way fragments are linked together. If the documents are updated frequently, they often share the fragments with static data content, whereas the fragments with dynamic full-text content are shared to a much lesser extent [RILD04].

3.2.2 Examples of XML Full-Text fragments

The test questions (Q1–Q3) might not be sufficiently exhaustive in all possible cases, so we want to complete the characterisation of XML Full-Text fragments with appropriate examples. Remembering that the scope of this thesis covers indexing and retrieving XML fragments when XML documents are too big or too unspecific to be indexed as a single unit, we identify three possible types of structures in the acceptable fragments:

1. A single element, possibly complex type content.
2. A range[2] of consecutive elements that is defined by the starting and ending point.
3. A set of elements that is not a continuous sequence in document order.

The first type is sufficient to represent all the indexed fragments if the answers to a query should function as starting points for user navigation. The second and the third type are useful when standalone-type answers are desired. In order for the third type to be meaningful, the elements have to be connected with links or by other ways. Although the definition allows for character data to appear before the start tag and after the end tag, in typical cases, character data occurs only between the start and end tags. Consequently, each entire fragment is a composite of whole elements of the source document.

[2] As defined in DOM Level 2, see
http://www.w3.org/TR/DOM-Level-2-Traversal-Range/

Besides the structural possibilities, XML Full-Text fragments may come in different sizes, too. The following list contains typical fragments representative of different granularities also including some that do not meet the requirements of the definition. The descriptions use the concept of *text depth* which is defined as the node distance between the fragment root element and character data. The numerical values originate in an experimental analysis of a large collection of scientific journals which is described in 6.1.

1. **Small inline-level element.** The interesting elements with text node siblings typically contain phrases of 1–20 words, e.g. conceptual terms, definitions, proper names, and quotations. On their own, they might not be of interest for the purposes of traditional information retrieval, but they do have potential in systems for Information Extraction and Question Answering. Even smaller inline elements are common, but character strings shorter than a word hardly meet the requirement of independence.

2. **Paragraph.** Most of the text (>90%) of the smallest block-level elements is stored in the child nodes of the root element which results in average text depths of 1.0–1.2. Paragraph elements have no text node siblings, but they may contain a single-digit number of descendant elements and 200–500 characters.

3. **Subsection.** Most of the text (>65%) of the smallest containers of block-level elements is stored in the grandchild nodes of the root element. The text depth averages around 2.2–2.8. Typical subsections contain 20–60 descendant elements and 1,000–3,000 characters.

4. **Section.** Less than 50% of the text is found in the grandchild nodes of the root element. The rest of the character data is deeper, which shows in the average text depths of 2.8–3.2. Sections typically contain 40–80 descendant elements, including subsection elements, and 3,000–7,000 characters.

5. **Traditional document.** In a stereotypical article, the distance to most of the text equals at most 5 nodes which results in average text depths of 4.0–5.0. The number of descendant

elements varies in the range 400–800 which is remarkably bigger than that of sections. The amount of characters show even more variance with counts between 10,000 and 100,000.

6. **Oversized document.** The documents that are too big to be retrieved as one unit include books, article collections, whole journals, etc. Most text nodes in these documents are at least six nodes away on the descendant axis starting from the root element. The number of descendant elements tops 10,000 and the number of characters may well be measured in millions.

Not surprisingly, we observe that the number of descendant elements of a fragment correlates with the average depth of the text nodes. We have assumed the Latin alphabet in the estimated character counts of the fragments, and the numbers may differ if the examples are taken from documents with a different character set. The size of unparsed entities is not taken into account here.

Whether the inline-level elements are at all suitable for indexing can be questioned for several reasons. For one thing, processing all inline-level elements is expensive. For another, if the content length is small, the element cannot be considered a unit worth indexing as a fragment of its own. Big inline-level elements may be a possible exception, however. For example, paragraph elements may contain list environments with enough text content to warrant the status of an XML Full-Text fragment, but in that case, we would still have to deal with orphaned text nodes as well as with having to prove the parenting paragraph element inappropriate for indexing.

The oversized documents form another class of fragments that cannot always function as XML Full-Text fragments. These big documents require suitable indexing methods that are capable of dealing with the structured nature of the content and not based on the traditional concept of a document. Such methods are not presented in this thesis.

3.3 Granularity levels

From the characterisations in Section 3.2, we proceed to more precise definitions. Two complementary definitions are introduced: 1) *evidential full-text fragments* where the evidence lies in the content

models and 2) *statistical full-text fragments* where the fragment size is the most important piece of statistical information. Both kind of definitions are applicable when modelling the granularity of full-text fragments.

The definitions in this section do not assume any element type (or name) for a number of reasons. For example, in full-text documents, elements of the same type have plenty of variation in their content. The variation is both structural and semantical, which is inherent to full-text, and it seems that the full-text elements — regardless of the element name — allow for arbitrary text to appear in the document. Moreover, the more the definitions generalise, the easier it is to apply them to heterogeneous XML documents, which is one of the important areas in this research.

3.3.1 Evidential full-text fragments

The purpose of the definitions for evidential full-text fragments is to get statistics concerning those fragments that are subject to full-text search (XML Full-Text fragments as defined in Section 3.2.1). With the statistics at hand, we are able to index and retrieve full-text fragments of a certain granularity that may not be specified in terms of the statistics but in natural language instead, such as "the granularity of sections". Obviously, the missing link in this process is a mapping between the human-readable label of the granularity and the computer-generated statistics originally collected from the indexed documents. That is where evidential full-text fragments enter the scene. Each one of them represents at least one level of the granularity which is specified by the user.

As the name suggests, we are looking for clear evidence of full-text content in this section. The challenge here and in the upcoming sections is to determine whether the content can be considered full-text or not in order to exclude other kind of XML which shall be called *data* in this thesis. XML fragments containing pure data are not considered reasonable answers on their own to full-text queries. They would also distort the statistics if treated as full-text XML. Recognition of full-text content is thus necessary. What adds to the importance is that full-text documents tend to contain other kind of text, too, which can usually be characterised as data, i.e. meta-data.

Because counting nodes and comparing their sizes does not necessarily make data and full-text look very different, we need to regard some stronger evidence in order to differentiate the two types of content. The definitions for evidential full-text fragments are based on 1) the content models of the root elements of the fragment and the ones of the descendant elements and 2) the minimum text depth which is the distance from fragment root to the nearest text node. With a DTD, five different content models can be defined: 1) Elements, 2) Text, 3) Mixed, 4) Empty, and 5) Any. Only the first three content models are meaningful when analysing fragment content. The mixed content model is the most challenging one to recognise as the elements may have the appearance of the Elements or the Text content models, as well. However, only those elements that have both text and element content have supporting evidence of full-text content, regardless of how exactly the mixed content is defined for the corresponding element type in the DTD. This reasoning originates in the key hypothesis:

Mixed content does not occur in elements that contain data.

Reasons behind the hypothesis are practical: Definining the mixed content model for data would make mapping XML elements to data items difficult. Moreover, as Michael Kay states, "*writing the code to handle the document would become a nightmare*" [WBD+00]. It is also a common practice to avoid the mixed content model when describing data. In this light, mixed content is a strong full-text indicator, which follows from the hypothesis.

We use DOM Nodes[3] and trees for distance calculation between the fragment root and the nearest Text node. Examples of common minimal distances are shown in Table 3.1. Our definition for a Text node corresponds to that of the DOM Text node, but we would like to exclude those with only whitespace content and those without any indexed content. Excessive whitespace is removed by normalising all Text nodes, but the removal of any non-empty Text nodes cannot be taken into account with XML tools only. In order to ignore the noisy content, we may remove stopwords, punctuation, and special characters, and possibly require a minimum length of the

[3]See, for example, http://java.sun.com/j2se/1.5.0/docs/api/org/w3c/dom/Node.html

Min. dist.	Granularity	Significant Text node axes
0	Character	Text node siblings
1	Paragraph	Text node children but no Text node siblings
2	Paragraph group or subsection	Text node grandchildren but no Text node children
n		Nearest Text node nth descendant

Table 3.1: Distances to Text nodes defined.

remaining content words. For the sake of completeness, CDATA-Section nodes are treated as Text nodes, as well.

The evidential full-text fragments are defined by the level of granularity which corresponds to the fragment size. Some of the names such as "inline level" are in common use while others describe the size of the fragment in a hypothetical document.

Level 0: Inline-level elements. The minimum text depth equals 0, which necessitates Text node siblings. The content model of the element itself is irrelevant, but the parent element must have the mixed content model. These elements occur commonly in full-text fragments and rarely in data fragments.

Level 1: Paragraph elements. Mixed content and a minimum text depth of 1 are required. The parent element must not have mixed content, which forbids the occurrence of Text node siblings. The Text node descendants must contain a sufficient amount of text, but setting the actual threshold is left as a matter of parameter tuning. Level 1 fragments are common in full-text documents and rare in documents containing only data as they are the parent nodes of Level 0 fragments.

Level 2: Subsection elements. Element content and a minimum text depth of 2 are required. At least one child element must be a Level 1 fragment. Level 2 fragments may contain other Level 2 fragments, but the content model of the parent element is irrelevant.

Level 3: Section elements. As at Level 2, element content and a minimum text depth of 2 are required, whereas the content model of the parent element is irrelevant. In addition, at least one child element must be a Level 2 fragment.

Level N: Indefinite elements. When N≥2, element content and a minimum text depth of 2 are required, and at least one child element must be a Level N-1 fragment.

Level A: Article elements. The coarsest granularity cannot be defined in terms of content models and text depths, but Level A fragments usually contain elements of all the Levels 0–3.

Definitions for Level 0 and Level 1 fragments are quite straightforward but the complexity increases towards the higher levels. For example, the elements that parent both Level 1 and Level 2 fragments are both Level 2 and Level 3 fragments themselves assuming that the other conditions are satisfied. This is not a problem as the definitions are used for statistical purposes only. Moreover, the descendant elements of Level 0 fragments can be Level 2 or Level 3 fragments because ancestor nodes are ignored in the corresponding definitions. However tempting it is to look into the ancestor axis, it would make the definitions weak as full-text fragments can have an unlimited number of ancestor elements by definition. Exact definitions for fragments bigger than sections are unrealistic in their complexity if only content models and text depth are considered because almost all the descendant elements can have any of the possible content models. This would deteriorate the quality of the fragments at Levels N where N>3.

It is not necessary that all full-text fragments match any of these definitions because they are used for statistical purposes only. Also, the amount of evidence of a fragment having full-text content varies from one fragment to another. However, we need a statistically significant number of fragments and a sufficient amount of evidence of the content being full-text.

Besides being independent of document types, the granularity levels also solve the problem of recursive structures where elements of the same type are nested, occurring at several levels. The statistics collected level by level are thus more accurate than those collected by self-defined classes of element types.

3.3.2 Statistical full-text fragments

Arbitrary XML fragments may not match any of the levels of evidential full-text fragments because of the lack of mixed content, which makes the definitions incomplete as a model for full-text fragments. A complementary approach is presented in this section by modelling the same granularities with corresponding statistical definitions which are simple and applicable to all full-text fragments. After several attempts to take advantage of different kinds of statistical information, we observe that measuring the size of the fragment is sufficient in order to determine the granularity of a full-text fragment. More statistical data including the average, minimum, and the most common text depth is available but discarded at the current stage of the research. Future work will show whether the other statistical information could help us determine the granularity of fragments and eventually determine the size of both the indexed and the retrieved units of XML. The definitions can be extended with the additional data at a later point of time if the value of adding complexity to the definitions becomes clear.

There are a number of different units for the size of a full-text fragment. For the sake of simplicity, the fragment size is measured in characters throughout this thesis. The length of the string value of the fragment is available through the DOM and SAX interfaces in the parsed XML document, and as a unit, the number of characters is independent of the document type. The statistical fragment size is defined by first measuring the evidential full-text fragments level by level according to the classification introduced in Section 3.3.1. If mixed content does not occur frequently in the document collection, the statistical size can be measured from other documents as long as the language and the character set are the same. Without a significant amount of evidential full-text fragments, the statistics are not representative of the full-text content in a whole collection of documents.

After collecting the statistics, we have the ingredients for the initial definitions for statistical full-text fragments. For example, for the paragraph level, we define the granularity using the following notation:

$$G_{paragraph} = \{s(Fragment_{L1})\},$$

where $s(Fragment_{L1})$ denotes the median size of evidential full-text fragments at Level 1. However, for most fragments, the size falls somewhere between the median sizes of different granularities, so we need to expand the definition into the form

$G = \{[min, max]\}$.

We assume there is a correlation between the average sizes of evidential and statistical full-text fragments. As our goal is to define a size range for each granularity, we need two correlation factors (cf_{min} and cf_{max}) with which the ranges can be defined. For example, the statistical definition for the size range of the section level is added to the corresponding definition for the granularity as follows:

$$G_{section} = \{[cf_{min} \times s(Fragment_{L3}), cf_{max} \times s(Fragment_{L3})]\}.$$

If we assume that the median size of a section element equals 5,000 characters, $cf_{min} = 0.5$, and $cf_{max} = 2.0$, we can define the corresponding granularity

$$G_{section} = \{[0.5 \times 5000, 2.0 \times 5000]\} = \{[2500, 10000]\}.$$

As the definitions are based on ranges on a gradual scale, we can actually model an infinite number of different granularities in addition to those defined in the previous section. This is useful when the granularity is not set by the user but by the requirements of the software or hardware instead.

Setting good values for the correlation factors requires a substantial amount of experimental testing with different size ranges in order to find the good minimum and maximum size for each granularity. The preliminary test results are included in this thesis but, unfortunately, no definite answers could yet be discovered. Once the ideal values are discovered, we can study the statistical distribution of fragment sizes at each granularity level and see if the ideal cf values can be derived from the distribution and if the same values are ideal for each of the granularity levels. Determining the actual values is left as a topic for future research.

Finding more sophisticated units for fragment size is another area calling for further research. For example, counting the actual index terms may lead to more precise retrieval results than simply counting characters. However, finding the index terms requires

extra parsing of the content. Moreover, counting characters was proven useful in related work: Ramaswamy et al. made a successful case of automatic fragment detection and defined the size of an *augmented fragment (AF) tree* as the length of the subtree value [RILD04].

3.4 Measuring the probability of full-text

Because not all full-text fragments qualify as evidential full-text fragments and because also other than full-text fragments qualify as statistical full-text fragments, we need a more precise definition that matches practically all full-text fragments and nothing but full-text fragments. As a solution, we study one of the contributions of this thesis: a way to classify XML fragments into full-text fragments and data fragments. An essential part of the classification is based on the structural clues in the fragment which are introduced in Section 3.4.1. How entity references can be taken into account in the measurements of full-text content is analysed in Section 3.4.2. Whether the measurements generalise to documents of an arbitrary document type is argumented in Section 3.4.3 in response to the critics who may have doubts. The definition of a statistical full-text fragment is extended into a more satisfactory form of a *qualified full-text fragment* in Section 3.4.4.

3.4.1 Full-text indicators

Given any XML fragment, we want to be able to make a judgment about the fragment containing a sufficient amount of full-text content. A sufficient amount of such content makes the fragment suitable for full-text search. An automatic judgment mechanism requires that we can measure the fragments on a scale where typical full-text fragments are situated at one end of the scale and typical data fragments at the other end. The important points are not at the ends of the scale, however, but in the middle where a *pivot point* divides the scale into ranges of full-text and data values. Indicators that return appropriate values on such a scale actually calculate the *Full-Text Likelihood* of the fragment. Based upon our earlier assumption about the meaning of mixed content, we propose that

the measures of full-text likelihood should reflect the proportional amount of mixed content in a fragment. We can identify two such measures which slightly differ from each other:

1. T/E (Text nodes/Element nodes) ratio. A pivot point between data and full-text is defined as T/E = 1.00 where a fragment with a T/E value greater than 1.00 is considered a full-text fragment.

2. C/E (Characters/Element nodes) ratio. The pivot point of the C/E measure is dependent on the language and the character set, but it maybe possible to derive the pivot point value from the document statistics. Full-text fragments generally have bigger C/E values than data fragments, which is similar to the behaviour of the T/E measure, but one undisputable pivot point cannot be determined.

The values of both measures fall in the range [0,∞] so that empty elements have the value of 0, data fragments have values in the range]0,p], and full-text fragments in the range [p,∞[, given that the pivot point p is defined.

In order to demonstrate the behaviour of the T/E and C/E measures, we study how elements with different content models measure. Example (3.1) shows an element with the "text only" content model where T/E = 1.00 and C/E = 9.00.

(3.1)
```
<p>text only</p>
```

This kind of elements are common in both full-text and data fragments, but by themselves, they are too small to qualify as a full-text fragment. The T/E value of 1.00 is typical of elements with only text content, and it does not indicate full-text or data content per se, but the very low C/E value of 9.00 is a strong indicator of a content other than full-text.

A typical case of a data fragment, where T/E < 1.00, is presented in Example (3.2).

(3.2)
```
<pubdate>
  <month>May</month>
  <year>1975</year>
</pubdate>
```

Observed model	Effect on a neutral element	Common content
Elements	decrease	data
Text	neutral or increase	data, full-text
Mixed	increase	full-text

Table 3.2: Content models in relation with the T/E values and types of content.

The interpretation of the fragment content is heavily dependent on the tag names such as "year" and "pubdate", which is typical of data fragments. The three Element nodes and two Text nodes result in a T/E value of 0.67 while the C/E value of the fragment equals 2.33. Both indicators clearly flag for the fragment containing data.

A fragment with the mixed content model is shown in Example (3.3). The T/E measure has the value of 1.50 which is expectedly on the right side of the pivot point of 1.00.

(3.3) `<p>Some <i>important</i> full-text.</p>`

The C/E measure has a rather low value of 12.50 because of the small size of the fragment. While the C/E value can be ten times higher in real-life full-text fragments as shown in Section 6.3, the T/E value of 1.50 is common in full-text fragments of all sizes.

The examples show how the T/E measure actually describes the content models of the elements of the fragment. The T/E value of a whole fragment combines the effect of the content models of the fragment root and its descendant elements. The possible effects of a single element on the T/E values are detailed in Table 3.2.

We define the pivot point value of 1.00 neutral to the type of content. Lower values which occur in elements that only have element children cause the T/E value of the whole fragment to decrease towards the range of data values, whereas mixed content has a similar effect of an increase towards the full-text values.

3.4.2 Entity references in element content

XML elements that have text content may also contain entity references to entities with string values. When parsed into a DOM Node, the text content with entity references consists of both Text nodes and EntityReference nodes, and, in data fragments in particular, the values of the T/E measure become deceptive. This phenomenon shall be called the *ER effect*. When measuring XML fragments, the ER effect shows in the average distance between the fragment root and the characters as well as in an increased number of Text nodes:

- A Text node child of an EntityReference node is deeper in the fragment than the Text node siblings of the EntityReference node although both appear at the same level when the fragment is parsed and the entity references are expanded.
- If a character in a Text node is replaced with an entity reference referring to the character entity of the character itself, the number of Text nodes increases by two. Afterwards, there are three Text nodes instead of just one: one preceding the entity reference, one following it, and the substituted character entity as a child node. If the replaced character is the leading or trailing character of the text node, only one new text node appears.

In particular, when the language of the text and the default character encoding (UTF-8) are not a perfect match, e.g. fragment (3.4), entity references commonly replace the special characters of the language. The T/E value of fragment (3.4) equals 5.00 whereas the C/E value equals 20.00.

(3.4)
```
<p>Text &agrave; la fran&ccedil;aise.</p>
      123456       789012345       67890
```

The T/E measure is sensitive to the ER effect whereas the C/E measure is not. We cannot, however, resort to using the C/E measure when entity references are common in data content, as the downside of the C/E measure is that determining the pivot point is not trivial. The textual size of a data item may fall in the same range as the size of a full-text fragment. Nevertheless, if we assume

that entity references are evenly distributed among data fields and full-text, the distinguishing capability of the T/E measure is not greatly influenced by the ER effect although the pivot point still requires adjustment.

If we want to eliminate the ER effect, we can modify the T/E measure into $\frac{T-2\cdot R}{E}$, where R stands for the number of entity references. As replacing a character in the middle of a text node increases the number of text nodes by two and the number of entity references by one, we have to multiply R by two. However, should R stand for all entity references with a string value or for only those referring to a single character is not clear. The original pivot point of the T/E measure is well defined for the modified measure, as well. For example, the value of the modified T/E measure equals 1.00 in Fragment (3.4).

3.4.3 Independence of document types

Whether the full-text indicators proposed in Sections 3.4.1 and 3.4.2 actually classify XML fragments correctly with appropriately defined pivot points should, without a question, be tested on authentic XML documents. However, a question can be raised about the quality of the test documents: How many document types need to be represented sufficiently in the document collection where the distinguishing traits — the full-text indicators and the corresponding pivot points — are verified? If we can show that the measures are independent of document types, one document type suffices in the verification as the validity in others can be induced, which is ideal. Otherwise, a wider variety of documents are needed in the test collection.

One argument against the independence of the document types says:

"*...some DTD might induce a high proportion of elements and some [might] not...*"[4]

According to our first counter-argument, a DTD cannot dictate that an element for full-text content parents a high proportion of elements. We reckon that the elements parenting full-text must

[4]From an anonymous reviewer of the ACM Fourteenth Conference on Knowledge and Information Management 2005 (CIKM'05).

```
<xsd:element name="fulltext">
  <xsd:complexType mixed="true">
    <xsd:sequence>
      <xsd:element name="a" type="xsd:string"
                   minOccurs="1"/>
      <xsd:element name="b" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Figure 3.1: Schema definition for a full-text element.

have either the "text only" or mixed content model. If elements are not allowed in the content model, even a single text node in element content makes the proportion equal to 1/1. When both elements and text are allowed, we need to consider the appropriate definitions. With the DTD, the mixed content model is defined as shown in Example (3.5).

(3.5) `<!ELEMENT fulltext (#PCDATA|a|b|c|d)*>`

There is no other way to define the mixed content model in a DTD. The `fulltext` elements as defined in the example may or may not contain text and the listed elements (`a`, `b`, `c`, and `d`) in an arbitrary order. The definition does not enforce the occurrence of any possible child element. The XML Schema definitions allow for more complex definitions for the mixed content model, though. An example is shown in Figure 3.1.

With a schema definition, the appearance and order of the child elements can be enforced in the mixed content model. The element `a` in Figure 3.1 provides such an example. However, only two content models in the descendant elements of the fragment may steer the T/E value towards the range of data values: the empty and "element only" content models. If the content model suggests the occurrence of several such elements, we may well question whether the defined element is truly designated to full-text content.

Another argument concerns the previously defined pivot point of the T/E measure:

"*...proposal sets an arbitrary value of 1.0 for the ratio of elements. ...different values may be appropriate for other collections (depending on the characteristics of the DTD) and no single value may be appropriate in a heterogeneous collection.*"[5]

In response to the arguments, we show that the pivot point value of the T/E measure comes from the content models of full-text elements, not from properties specific to any DTD, and thus, the value of 1.0 is not chosen arbitrarily nor is it dependent on the characteristics of a DTD. If only text content is allowed, the T/E value is at least 1.00 depending on the number of entity references. If mixed content is allowed, the occurrence of any elements cannot be enforced with a DTD. Moreover, requirements for elements with the "element only" content model is not meaningful — it even contradicts with the synonymous concept of *free text* — although it is possible with a schema definition.

With the pivot point of 1.00, single elements can be misjudged by only looking at the T/E value but all full-text content cannot. It is always possible to have a full-text element with a T/E value of at least 1.00. One of the few cases where full-text content could be misjudged as data is when some structured content, such as a table, occurs at the inline level. Another such case occurs if all the words of the full-text are wrapped in individual elements, e.g. if the text is tagged with elements marking the part-of-speech of the words. Judging these borderline cases as data might be a very reasonable choice of action. Data content can be misclassified, too, if mixed content is allowed in data elements but again, instead of questioning the pivot point value, we may as well question the content being data.

There are also cases which may at first sight seem like full-text documents but, as expected, the appropriate full-text indicators reveal the real quality of the content. A typical example of such a document is shown in Figure 3.2 where the content cannot be considered data in the sense it has in the context of databases. However, not all the criteria for an XML Full-Text fragment are met, either, as the interpretation of the content is heavily dependent

[5]From another anonymous reviewer at CIKM'05 conference.

```
...
<SCNDESCR>SCENE  England; afterwards France.</SCNDESCR>
<PLAYSUBT>KING HENRY V</PLAYSUBT>
<ACT><TITLE>ACT I</TITLE>
 <PROLOGUE><TITLE>PROLOGUE</TITLE>
  <STAGEDIR>Enter Chorus</STAGEDIR>
  <SPEECH>
   <SPEAKER>Chorus</SPEAKER>
   <LINE>O for a Muse of fire, that would ascend</LINE>
   <LINE>The brightest heaven of invention,</LINE>
   <LINE>A kingdom for a stage, princes to act</LINE>
   <LINE>And monarchs to behold the swelling scene!</LINE>
...
```

Figure 3.2: Excerpt from the Shakespeare play *Henry V* marked up in XML, courtesy of Jon Bosak.

on the tag names, and the fragment has to be seen as data in the binary classification.

3.4.4 Qualified full-text fragments

The definitions for the granularity of a full-text fragments in Sections 3.3.1 and 3.3.2 offer a good basis for further extension, and they will be reused in this section as follows. First, we analyse the classes of evidential full-text fragments at different levels of granularity. Second, we define the corresponding statistical full-text fragments by using the results of the analysis. Third, we restrict the definition with a requirement for full-text content because the statistical definition is not strict enough by itself. The additional requirement is supposed to help discard the fragments that do not have full-text content so that only *qualified full-text fragments* fit the definition. The restricted definition for a statistical full-text fragment includes the requirement for full-text likelihood. In general, the granularity of qualified full-text fragments is defined using the following notation:

$$G = \{[min, max], T/E \geq 1.0\}.$$

For example, the paragraph granularity is defined as follows:

$$G_{para} = \{[cf_{min} \times s(Fragment_{L1}), cf_{max} \times s(Fragment_{L1})], T/E \geq 1.0\}.$$

Assuming that the median size of a paragraph-level fragment equals 500 characters so that $s(Fragment_1) = 500$, $cf_{min} = 0.3$, and $cf_{max} = 5$, the paragraph granularity is then defined as

$$G_{paragraph} = \{[150, 2500], T/E \geq 1.0\}.$$

A requirement for a C/E value may be added to the definition when the pivot point is well-defined. While we are uncertain about the cf values and whether they are dependent on the level of granularity, we know that the same pivot points of the T/E and C/E measures can define any granularity as they describe the full-text likelihood of the fragments instead of measuring the fragment size.

With the definitions presented in this section, we can determine whether any given XML fragment is a qualified full-text fragment at a given granularity. How the definitions are used in practice is shown in Chapter 6 where a whole document collection is analysed.

CHAPTER 4

Selection of indexed fragments

While Chapter 3 introduced individual XML fragments in the context of their source documents, our interest in this chapter centres on whole collections of XML fragments and methods for dividing XML documents into such fragments that are indexed for full-text retrieval. Defining the properties of an ideal selection of fragments is far from trivial in the scope of heterogeneous XML documents. Nevertheless, the goals regarding the content of a fragment collection are skecthed in Section 4.1, followed by the structural issues concerning the relation of individual fragments to the source documents and to each other in Section 4.2. The differing characteristics of data and full-text and the consequent effects on dividing XML documents into fragments are the topic of Section 4.3, whereas Section 4.4 is devoted to the algorithm that determines which fragments are indexed and how they are selected. Finally, metrics for evaluating the qualities of the division process and its outcome are investigated in Section 4.5.

4.1 An ideal collection of fragments

Collections of indexed documents traditionally consist of documents in their entirety with quite natural boundaries between each document. Taking into account that we want the traditional methods to be applicable to our XML fragments, the set of indexed fragments should resemble the traditional collection to some extent. For example, the text content of an XML fragment should correspond to

that of a document in its traditional sense so that language models and vector spaces can be defined in terms of these fragments as if they were independent documents. Properties that are obvious in the traditional documents must, however, be carefully defined for the fragments which can be seen as virtual documents inside XML documents before they are materialised into actual "documents" that are returned to the user. Our wishes regarding the content of the fragment index are presented in Section 4.1.1 followed by those for the quality of the individual fragments in Section 4.1.2.

4.1.1 Content to be included

The goals for the contents of the final fragment collection have to be explicitly defined because not all of the content in the original documents is worth indexing. The process of dividing documents into fragments is also a good opportunity to filter out some unwanted content which might only be detected by analysing the XML structures associated with the content. Because the goals indentified in this section are hypothetical, they are not intended to be completely exhaustive regarding the ideal fragment collection. Instead, they will give motivation and direction to the methods presented later on in this thesis.

There are two approaches to setting these goals. First, we specify which content should be included in the index, and second, we characterise what could be excluded.

Full-text All of the full-text content should be included by default. This goal is fundamental as we are developing methods for full-text search. Heavy arguments are required if any full-text content is to be ignored.

Meta-data The meta-data describing the fragments is associated with the fragments in the source documents. The associations should be preserved in the fragment index. The important meta-data includes titles, captions, keywords, and even footnotes. The importance of the meta-data is magnified in relatively small fragments with little amounts of full-text content.

Link relations A link connecting two elements is broken if the elements end up in different fragments in the fragment collection. The link itself can be lost, but the information it codes for should be preserved. Such information includes the content at the other

end of the link.

Exclusion of unimportant content In principle, text that is not likely to be a satisfactory answer to any full-text query can be discarded. Such text can hardly serve as meta-data, either, which further supports its exclusion. Common examples include elements containing no index terms, elements with no indexed content, or elements with an excess of only index terms and page numbers, e.g. a fragment containing a keyword index. However, this goal with its vague definition is not crucial because good indexing techniques are able to handle the noise that is not filtered at this point.

None of the goals presented here are applicable to traditional documents where all the text of the undivided text documents is included in the index by definition, and meta-data as well as link relations can only be lost if the documents are divided into fragments. If so desired, unimportant and noisy content can be cleaned at the same time as punctuation and stopwords are removed upon the application of language models. In order to achieve these goals for XML documents, we have to succeed in detecting the appropriate content in the original document collection, after which it can be included in the set of indexed fragments.

4.1.2 Fragments as individual entities

So far, we have identified two major requirements for the indexed fragments regarded as individual entities: the fragments should be both *coherent* and *independent*.

Coherence Topical unity is ideal in the indexed fragments. It shows in that a coherent fragment is almost completely either highly relevant or highly irrelevant with respect to a given query, and in that it never needs to be divided further into smaller fragments. In comparison, all the seemingly similar items in a bibliography are not as closely related as consecutive paragraphs in a section, which explains the incoherence of bibliographies when compared to coherent sections. Other incoherent fragments that should not be indexed as independent units include tables of contents, indices, glossaries, and collections of articles.

Independence Each fragment should be independent of other fragments including those that are not indexed. An independent fragment can be indexed as a single unit, and it can be included

as such in the list of retrieved answers for a query. Moreover, information outside the fragment is not needed when the relevance score of an independent fragment is computed. For example, it has no significant links pointing to elements outside the fragment, and as stated in the previous section, we consider all links significant.

Bibliographies are stereotypical examples of document fragments that fail to meet either of the requirements. They are dependent on the citations and they may deal with a wide variety of different topics. In practice, the significance of rejecting bibliographies as individual fragments shows when the search terms occur frequently in bibliographical data and rarely in the full-text content, i.e. "transactions", "journal", "proceedings", and "communications". These words can actually stand out in the other fragments even when they are less common there than in bibliographies.

The qualities of coherence and independence are highly appreciated in undivided documents, too, but the only way towards meeting these fine qualities is to reject the unacceptable documents in their entirety. When the documents are divided, we have more choices at hand: we can adjust the parameters that control the division process and we can also reject particular parts of whole documents. In this thesis, we choose an approach where we first define a set of coherent fragments to be further refined and expanded so that the goal of independence is achieved, as well.

When the goals for the structure of individual fragments are considered at the collection level, it is not realistic to expect that 100% of the indexed fragments meet the requirements. An ideal collection of fragments is likely to contain some fragments of borderline quality, because in some cases excluding them would lead to a bigger loss than including them in the index.

4.2 Structural issues

Dividing a document collection into fragments with unsupervised methods hardly results in an ideal fragment collection, but it is reasonable to expect the result to be a good compromise. There are different ways to achieve this goal depending on the choices that are made about how the fragments are related to whole XML documents.

Overlapping fragment context information For example, all parts of a section are under the same section title which belongs to the fragment context. Several fragments linking to the same element is another typical example of context information that is shared by multiple fragments. If the link relations are preserved, the common element should be duplicated several times. Otherwise, the fragments are not independent. Terms appearing in the copied elements may seem more common than they really are, which in some weighting schemes leads to lower term weights. This should be taken into account when duplicating content.

Partly overlapping fragment bodies When a section with three subsections is divided into two fragments, we may choose to include the second subsection in both fragments. In addition to the shared subsection, both fragments also contain a non-overlapping subsection. Even partial overlap complicates the process of creating indices, and anyway, overlap is an undesired property per se in the result list for a query.

Nested fragments The special case of overlapping fragment bodies is such that one fragment is completely included in another as a descendant fragment that comprises at least one descendant element. Nested fragments increase redundance in the index, unless the original document structure is preserved and the overlapping parts of each fragment are only indexed once. The attempt to avoid redundance would, however, compromise the goal of keeping the fragments independent. Another factor against nested fragments is that although the granularity of returned answers should be determined dynamically, e.g. based on their topical relevance, it does not imply that the size of the indexed units could not be fixed. As a conclusion, we can safely choose not to have nested fragments in the index.

Discontinuous fragment bodies Full-text fragments may contain sections of data that are not needed in the index, e.g. data charts of listings of program code. Removing the data is rather straightforward, but it is still unclear whether we have good reason to deny access to the data sections when the stripped fragment is given to the user. Returning a node or a location in the original document is challenging, should part of the corresponding subtree be excluded.

Fragment bodies with several root elements When a big element has a flat structure, it typically contains many small child elements or many text node children. The most natural way to divide such an element into fragments is to define split points where one fragment ends and another one begins. All the root elements of the resulting fragments have a common parent element which is not, however, part of the fragment. Fragment segmentation is an additional challenge that is avoided when the element structures are not flat.

Fragment bodies with orphaned text nodes In the rare case that a block-level element is too big to be indexed as a single unit, we may have to divide the element into fragments that start and end with either text or inline-level elements. However, block-level elements are rarely so big that we would lose significant amounts of full-text by requiring that all text nodes have a parenting element node in the same fragment.

The decisions concerning the structural issues do not have to be black-and-white yes or no answers, but we can also define a degree in the grey area, e.g. describing how much overlap and nesting is acceptable. Independent fragments can overlap and they can be nested, but the combination at a later point of time may be more efficient with non-overlapping fragments.

Any overlap also causes interference with the concept of *document frequency (df)*. For example, if a term occurs exactly once in two different documents and zero times elsewhere, then the term occurs exactly twice in the document collection and the document frequency of the term equals 2. However, if the documents overlap and if the term occurs in the common part of the overlapping documents, it only occurs once in the collection, thus contradicting with the *df* value. This issue is discussed in more detail in Section 5.3.

The solutions to these issues often come from the features and limitations specific to the implementations that assume certain user models, e.g. ones where standalone answers are required, or others where pointers to relevant nodes in the document tree are sufficient. Overlap of the answers is acceptable when they are treated as starting points for user navigation, whereas overlap in standalone answers causes redundance. Moreover, if all the indexed fragments are potential answers to users' queries, the format and presentation of the answers may set requirements for the structure of the in-

dexed fragments. For example, discontinuous fragments cannot be presented with standardised pointer or linking techniques, and in some cases, also fragments with multiple root elements are beyond the expressive power of the result list format. The structure of the indexed documents is another factor that may not leave us many choices concerning the structural issues of the indexed fragments, in particular when the structure is either very flat or deeply nested.

4.3 Discarding data fragments

When the indexed fragments are selected from a collection of documents, an appropriate size is the most important requirement for the fragments. However, granularity that is purely defined by the size is not the only factor when selecting the fragments, but also fragment content should be taken into account. As a contrast to Section 3.3.1, our perspective in this section consists of a whole collection of full-text fragments instead of single fragments. Despite that an ideal collection contains as few data fragments as possible, we may still include all the contents of the data fragments in the collection of full-text fragments as is proposed in Section 4.3.1. In Section 4.3.2, we will see how different kinds of content are treated differently and how setting limits on the minimum and maximum size of fragments is not enough.

4.3.1 Data and full-text separated

XML documents have traditionally been divided into data-centric and document-centric documents. One of the earliest references to the term *document-centric* dates back to June 1997 when Bruce Peat contrasted databases and full-text search engines on the xml-dev mailing list[1]. In 1998, the term *data-centric XML* became commonly used on the same mailing list. A corresponding division can be done to the content inside one XML document. To avoid confusion, we talk about *full-text-oriented* and *data-oriented* XML instead of using the terms associated with different types of whole

[1] http://lists.xml.org/archives/xml-dev/199706/msg00001.html [6/6/2005]

XML documents. The separation is based upon the following hypotheses characterising the differences between data and full-text.

Hypothesis H1 Data-oriented content has a coherent structure, but content-wise, it is incoherent in comparison with full-text content. Typical data-oriented content includes lists of references, lists of tables, figures, and footnotes.

Hypothesis H2 Data-oriented content is more dependent than full-text-oriented content on content in other fragments. For example, bibliographic data is included only when referred to.

In addition to the separation of data and full-text at the conceptual level, we perform the separation in practice. The motivation comes from hypotheses H1 and H2, according to which XML data fragments are incoherent and dependent on other fragments, which makes them less than ideal to be included in the fragment collection. Data may also appear as noise if data fragments are indexed together with full-text fragments, e.g. if the keyword index of a thesaurus is indexed as an individual entry.

When data-oriented content is associated with references in full-text-oriented content, the separation causes portions of the referring content to be separated from the referred data. Examples include references to the bibliography or out-of-line content such as footnotes. After the referred data-oriented content has been separated from the documents, it is completely excluded from the fragment collection unless it can be reassociated with the selected full-text fragments, e.g. through ID-IDREF relations.

A successful separation of data and full-text results in a fragment collection where all the fragment bodies are full-text fragments and data can only be found in the fragment context information. Although some data-oriented content might not be included in the index at all, it can still be returned as an answer to a query. For example, the data-oriented parts of an article should not be indexed as independent units, but they are included in the answer when the whole article is returned, regardless of whether they were included in the index or not.

4.3.2 Criteria for separating full-text from data

The names of XML elements are one of the simplest criteria for separating data from full-text. However, as the tag names are de-

pendent on the vocabulary associated with some document type, we have to discard the criterion as unadaptable to heterogeneous documents. The second most obvious criterion could be the size of the element, but then the argument is that XML elements of equal size are rarely of equal interest. Let us compare the stereotypical back matter element `<bm>` and section element `<sec>`. Although both of them have the same average size of around 5,000 characters, full-text is more likely found in section elements than in back matter elements. Unfortunately, this sort of behaviour is only typical of elements of these types and it cannot be taken as a rule. Exceptions are common, such as data-oriented sections and back matters with full-text content. More information is thus needed in order to reliably distinguish between a full-text element and a data element.

Assuming that the mixed content model implies full-text content, we can base our distinctive criteria on the definition of a qualified full-text fragment which was presented in Section 3.4.4. According to the definition, when the number of elements in the fragment content is greater than the number of text nodes, the fragment is considered data-oriented. By measuring the full-text likelihood, we can identify and discard data fragments without any explicit knowledge about the element names or types. If the division into fragments is based on the criterion of sufficient full-text likelihood, we can safely assume that the resulting fragment collection consists of full-text fragments.

4.4 Algorithm for fragment selection

So far, we have considered what kind of fragments are preferred in the index, how they are identified, and what kind of properties a collection of such fragments can have. At this point, we are ready to present an algorithm that determines from heterogeneous XML documents a set of XML Full-Text fragments of a chosen granularity.

4.4.1 Requirements

In Section 4.2, we looked at various structural issues concerning the output of the algorithm. As we have to account for the issues prior to developing the algorithm, the choices we make are treated as requirements for the algorithm in this section. Firstly, non-nesting fragment bodies are a necessary requirement because we highly value the independence of fragments and because we want to take advantage of the traditional IR methods which assume that the documents are independent of each other. Secondly, we allow overlapping fragment context information so that we can experiment with different ways to deal with the overlap. We are also interested in the significance of the problem on the whole: Are the overlap-originating changes in term and document frequencies so unevenly distributed over the fragments that the retrieval precision is jeopardised? Test results both with and without overlapping context information are presented in Chapter 8.

Because of the INEX-related limitations in our test environment, each fragment must consist of a single root element, which in practice disallows partial overlap, discontinuous fragment bodies, and orphaned text nodes outside the fragment root, when returning answers to queries. Whether we still allow discontinuous (or incomplete) fragment bodies in the index can be reconsidered later without having to make substantial changes to the algorithm. These limitations may be lifted in the future INEX initiatives, which would give us more freedom in the design of the algorithm.

One more requirement comes from the definition of the scope of this thesis. As we want to be able to index heterogeneous XML documents, the tag names as well as other information specific to a document type must be disregarded. The definitions of evidential, statistical, and qualified full-text fragments already meet this requirement, so there should be no major problems in developing the algorithm for selecting full-text fragments along the same lines.

4.4.2 Parameters for the algorithm

The granularity level of the indexed fragments is the most significant parameter for the algorithm. In a user-centred scenario, the level of granularity could be one of the user preferences which is

presented verbally, e.g. paragraph, subsection, section, or article. Because the algorithm requires the parameters in a more precise form, the name of the level is interpreted in terms of limits for the size, as well as other significant factors. For example, the granularity can be verbally presented in the form

$$G = G_x, \text{ where } x \in \{paragraph, subsection, section, document\},$$

which is converted into a size range for full-text fragments, so that

$$G = \{[min, max], T/E \geq 1.00\}.$$

When the interpretation of the names of the granularities is based on a mapping between the names and the levels of evidential full-text fragments, e.g. paragraph $\simeq$ L1, the minimum and maximum sizes can be inferred from the collection-specific statistics as shown in Section 3.3.2. It is nonetheless irrelevant to the algorithm how exactly the parameter values are set, as long as the format is precise, e.g., the verbally given name of the granularity can be interpreted.

Given granularity G, we perform the division d of a document collection C into n fragments f_1, ..., f_n. The division can also be defined as a function

$$d_G(C) = \{f_1, ..., f_n\},$$

where C is a set of arbitrary XML documents. According to the requirements, the bodies of the fragments f_1, ..., f_n must be disjoint, but they may share context information with each other.

Additional parameters concerning the postprocessing of the selected set of fragment bodies are presented in Chapter 5 as they are irrelevant to the algorithm that merely determines the composition of such set.

4.4.3 Tree traversal

As each of the divided XML documents is considered independent of the other documents, we process the collection of documents serially, one XML document at a time. Because the definitions

for different kinds of full-text fragments require that we operate on document trees, the algorithm basically defines how the tree is traversed and what to do with the traversed nodes.

The naïve approach is to traverse the whole tree from first to last node and test all the element nodes on the way. Besides being slow, this approach may also lead to other problems concerning nested elements because it is possible that both the parent and the child elements qualify. However, being thorough may work if overlapping and nested fragments are acceptable in the index, but otherwise, additional testing of nodes is necessary for each node that is traversed. If we optimise the tree traversal by adding conditions on branch selection we do not only make the traversal more efficient but we also have the option of restricting the output to disjoint fragments without an extra cost.

In the optimised approach, the beginning state of the division consists of an empty fragment collection F = $\emptyset$ and the current node c which is set to be the document root. The current node is first tested for size. If the size is bigger than the maximum size of the granularity, c is assigned each of the child element nodes one by one. If the size of the current node falls in the given range, the node is tested for full-text likelihood. In the case that c has too structured content so that T/E<1, c is again assigned each of the child element nodes. Otherwise, when the size of c fits the given range and when the content passes as full-text, we add c to F and move on to the next branch (subree sibling) in the document tree. The acceptance as a full-text fragment and a size below the given range are both stopping conditions for the recursive testing of child nodes. In other words, the child nodes are tested until they contain enough full-text or until they are too small. The tree traversal is then repeated for each document in collection C. When the algorithm stops, the fragment bodies that have been added to F represent the disjoint full-text fragments that can be selected from the XML documents in the colletion.

The pseudo code for the algorithm is presented in Figure 4.1. The accepted size range [Gmin, Gmax] is considered a global variable more than an actual parameter by its nature, and only the current node c is passed on at each recursive function call. The algorithm can be optimised by reordering the tests. However, the optimal order is dependent on the input documents. For example,

```
Algorithm SelectFullText(node) {
   if size(node) > Gmax
      // Discard node (too big)
      for each child: SelectFullText(node.child);
   else if size(node) < Gmin
      // Discard node (too small)
      break;      // Skip children
   else if ftl(node) = data //Discard node as data
      for each child: SelectFullText(node.child);
   else
      accept(node);
      break;
}
```

Figure 4.1: Pseudo code of the algorithm that returns a set of disjoint qualified full-text fragments from a given XML document.

if the test for full-text likelihood `ftl(node)` fails more often than the test for the fragment size, it should come before the size test assuming that they can be computed at an equal cost.

If a relatively small maximum size is given to the algorithm, the conditions for testing the child elements should be reconsidered. Small elements are likely to have text node children, from which follows that the child elements are at the inline level. Although some inline-level elements are seemingly good candidate fragments, e.g. lists, tables, and their inclusion in the division is easily justified, we are bound to discard their text node siblings, given that disjoint fragments are required. Whether this is generally acceptable depends on the nature of the documents and what would actually be lost in the orphaned text nodes. As a solution for these cases, we can define a soft limit for the maximum size of a fragment. Elements that have text node children are then tested against the soft limit instead of the actual maximum size. This solution compromises the definition of a qualified full-text fragment, though.

The problem of determining the set of full-text fragments is actually treated as a classification problem in the algorithm. It classifies

all the nodes of a document into three pre-defined classes: 1) Qualified full-text fragments, 2) "Too small" fragments, and 3) Other fragments (which are either too big or too data-oriented). Although the classes are fixed, the classification rules must be learned if no supervision is available. The unsupervised classifier is trained by collecting statistics from the document collection itself before it can be used in the algorithm. How to learn the minimum and maximum size for each granularity level was studied in Section 3.3, and some experiments with an actual document collection are reported in Chapter 6.

4.4.4 Example articles divided into fragments

In order to demonstrate how the algorithm works, we give two example documents to the algorithm and study the details of both the tree traversal and the output. The granularity is defined as

$$G = \{[150, 8K], T/E \geq 1.00\}.$$

The first input document represents a typical small article extracted from a collection of scientific articles. The file size equals 19,888 bytes. When parsed into a DOM tree, the XML document contains 204 Element nodes and 17,061 Characters in 345 Text nodes. The Attribute nodes and their values are ignored as they are not needed in the algorithm. The first element to be tested for size is the `article` element which is bigger than the upper bound of 8,000 characters. Then we proceed to the child elements and test each of them for the size as shown in Figure 4.2 which also shows the document order, according to which the tested elements are processed.

A total of 15 out of 204 elements (7.4%) in the small article is tested for size. The ten elements whose size falls in the accepted range are also tested for full-text likelihood. With the chosen parameters, the division of this article results in ten disjoint full-text fragments. The front matter (`fm` element) which typically contains the abstract of the article did not qualify because of its small size as there was no abstract in this article. Another less typical feature in this article is the back matter (`bm` element) which just barely qualifies as a full-text fragment. While most back matters seem too

```
          PATH              T/E    SIZE [150,8K] T/E>1.00
/article[1]:                1.69  17061    Too big
/article[1]/fno[1]:         1.00      5    Too small
/article[1]/doi[1]:         1.00     19    Too small
/article[1]/fm[1]:          1.10    114    Too small
/article[1]/bdy[1]:         2.36  15499    Too big
/article[1]/bdy[1]/sec[1]: 1.57    265    FT-qualified
/article[1]/bdy[1]/sec[2]: 2.25    971    FT-qualified
/article[1]/bdy[1]/sec[3]: 1.71    822    FT-qualified
/article[1]/bdy[1]/sec[4]: 1.71   1967    FT-qualified
/article[1]/bdy[1]/sec[5]: 2.00   1844    FT-qualified
/article[1]/bdy[1]/sec[6]: 3.33   2248    FT-qualified
/article[1]/bdy[1]/sec[7]: 2.63   3959    FT-qualified
/article[1]/bdy[1]/sec[8]: 2.80   1119    FT-qualified
/article[1]/bdy[1]/sec[9]: 1.88   2304    FT-qualified
/article[1]/bm[1]:          1.03   1424    FT-qualified
/article[1]/bm[1]/bib[1]:   0.81    626    (not tested)
/article[1]/bm[1]/app[1]:   2.50    798    (not tested)
```

Figure 4.2: Paths, T/E values, and sizes of the interesting elements of the small article.

data-oriented, this one is exceptional for two reasons. First, it has relatively few bibliographical entries, and second, it ends with an appendix containing the biography of the author.

While the algorithm nicely adapts to these less typical features by discarding the front matter and by accepting the back matter element, it also reveals its weakness by including the bibliography (`bib` element) inside the qualified full-text fragment. We may think that the problem goes back to the pivot point value of 1.00 which in this case seems just a little too low, but when half of the fragment consists of data and the other half of full-text, any T/E value near 1.00 actually describes the fragment perfectly. The criticism should thus be directed at the algorithm itself. By increasing the number of tested nodes, we are able to detect these fragments that are half data and half full-text, but in practice, the solution complicates the algorithm so much that the issue is dropped to the bin of ideas for future work on the topic.

The second input document with a file size of 251,158 bytes is one of the biggest articles in the whole document collection which is described in Section 6.1. The document consists of 4,635 elements, 5,384 Text nodes, and 182,497 Characters in the Text nodes. Altogether, 1,106 elements (22.9%) are tested for size. Out of the tested elements, 13 elements (1.2%) are too big, 934 elements (84.4%) are too small, and 159 elements (14.4%) are tested for full-text likelihood. The number of data-oriented elements equals 97 (8.8%) whereas only 62 elements (3.3% of the tested elements, 1.3% of all elements) are finally accepted as qualified full-text fragments. A more extensive listing of the tested nodes is included in Appendix A.

By looking into the bibliography at the end of the article, we see how the algorithm systematically discards the bibliographical entries. All of the 201 bibliographical entries were tested for size. The number of entries that were discarded as too small equals 104, whereas 96 out of the 97 entries that were big enough were discarded as too data-oriented. Only one bibliographical entry with the size of 180 characters and a T/E value of 1.00 (T=10, E=10) passed both tests, thanks to the character references in the text content. The element classification accuracy still reached 99.5% for the entries in this extensive bibliography.

The keyword element (`kwd`) in the front matter is another example of a fragment misclassified as full-text although its content

is much less data-oriented than that of a bibliographical entry. Example (4.1) shows how the element content starts with a `b` element containing a title-like phrase, after which the character reference separates the `b` element from the remaining comma-separated list of keywords. The size of the keyword element equals 328 characters, and the full-text likelihood T/E = 3/2 = 1.50.

(4.1)
```
<kwd><b>Index Terms</b>&mdash;Review, content
based, retrieval, semantic gap, sensory gap,
narrow domain, ...
</kwd>
```

The keyword element demonstrates a weak point in the T/E measure rather than one in the algorithm. As hinted in Section 3.4.3, fragments with unstructured data may look like full-text fragments if only the T/E value is consulted. The algorithm, however, handles nicely the front matter element (`fm`) which itself does not qualify as too data-oriented (T/E value of 0.98), but the abstract inside the front matter does (T/E = 1.66).

Altogether, we detected 10 qualified full-text fragments in the small article and 62 in the big one. Most of them seem to pass the tests for XML Full-Text fragments introduced in Section 3.2.1, but some clearly do not. Besides the bibliographical entry, it is hardly meaningful to retrieve the keyword element as one unit because it only contains a comma-separated list of noun phrases. In addition, the interpretation of the fragment content would be difficult without knowing the tag name which in this case tells us that the content is actually metadata about other fragments. Another questionable fragment is the back matter element in the small article which contains both bibliographical entries and a biography of the author. The education and awards of the author together with the literature that he cites make the fragment rather incoherent in comparison with the other qualified fragments. In addition to the two example documents, we study in less detail how a whole collection of documents divides into fragments in Chapter 6.

4.4.5 Finding the maximal number of fragments

Oversized elements and a flat structure is a combination that the algorithm has problems with: If the size of an element is bigger than

the upper limit and if the sizes of all child elements are smaller than the lower limit, no elements in the subtree qualify. This is typical of big elements with the mixed content model which shows in numerous text node children and inline elements.

One possible solution involves the modification of the algorithm by introducing soft limits in addition to the size range given as a parameter. The soft limits would be in effect only when no fragment in a subtree passes the size test. Those new fragment bodies that can now be selected still consist of a single element. This solution is not perfect, however. Defining the soft limits is not trivial, and even when they are defined, we can find elements that do not fit in the range between the soft limits. Nevertheless, making the upper limit flexible might solve the problem in practice despite its complicating effect on the algorithm where additional testing of nodes would be required.

Another approach to solving the "flat structure" problem is to look at the fragments that are discarded as too small. Those that form contiguous sequences should first be tested for full-text likelihood, after which they can be grouped into fragments of an appropriate size. The combinations of discarded elements that meet the requirements of a qualified full-text fragment are potential candidates to be included in the index. The compromise in this solution is that these additional fragment bodies have more than one root element, which has to be taken into account. An additional challenge may be caused by the small size of the contiguous sequences of discarded fragments. The fragments that are big enough may break the flat structure into such pieces that the remaining content does not include any contiguous sequence that passes the size test.

How to ensure the inclusion of all the important content in the fragment index when XML documents are divided into disjoint full-text fragments is still an open problem, though not significant enough to deserve more attention in this thesis. In the current setup, only marginal amounts of full-text are discarded when the granularity is properly defined. Referred content, out-of-line content, and other data can be added to the fragment index later if necessary.

The significance of the lower size limit for the qualified fragments may be questioned so that also the very small fragments are indexed, even when they only contain a few words. From the full-

text point of view, however, small fragments are unnecessary. A single noun phrase is around the size of a typical full-text query that consists of keywords and keyphrases. A perfect match would be identical to the query, but an identical match is of no value to the user. Also, the distinction between data and full-text is blurred when the fragment size is measured in numbers of phrases. Although systems for Information Extraction and Question Answering can benefit from small answers such as single noun phrases, the lower bound for the fragment size is an important parameter when full-text fragments for Information Retrieval are in question.

4.5 Evaluation of fragment selection

No standard evaluation metrics exist for evaluating algorithms for fragment selection. In this section, we consider the possibilities for developing such metrics so that we can compare the behaviour of the algorithm when it is given different parameters. As the algorithm is run offline, we will not grade its effectivity — how soon it stops — but we will evaluate the output instead by analysing the set of selected fragments. Because we look at the resulting fragments, the evaluation is dependent on the set of input documents. An ideal algorithm might not adapt perfectly to all possible document collections, but it should produce decent results with at least most of them. The properties that can be measured are regarded in Section 4.5.1. Two articles are again taken as examples in Section 4.5.2 where the evaluation results are reported as far as they are applicable to the division of single documents. Finally, in Section 4.5.3, we consider the evaluation of optimal results in order to give direction to future efforts on the development of better algorithms.

4.5.1 Measured qualities

Two kinds of measures are identified for the evaluation of an indexed fragment collection. The absolute measures describe the relation between the original document collection and the fragment collection which is the result of the division. The relative measures reflect the performance of the fragment collection with certain IR methods and a certain set of test queries.

The absolute measures include two types of metrics for coverage: one where a 100% coverage is trivial to achieve but far from the actual goals for fragment selection and another where the 100% coverage is the actual goal. The first type is related to the size of the fragment collection in proportion with the size of the original document collection. We can measure the size in different units, for example, as the number of Characters included in the collection divided by the number of Characters in the original collection. Other countable units include bytes[2], words, index terms, Text nodes, and elements.

The definitions for second type of the absolute measures are close to our goals for the output of the algorithm, which is why measuring the subjective error rate of the algorithm itself indicates how far we are from the optimal coverage. For example, we can study how many fragments containing full-text were misidentified as data so that it led to the discarding of the full-text, or we can detect how often data was misidentified as full-text. Being difficult to measure automatically, the error rate may have to be measured as a proportional size instead. If we assume that elements with the mixed content model are full-text fragments, the error rate is measured as the proportion of mixed content that was not included in the fragment collection. However, this strict interpretation of the mixed content may lead us astray in the evaluation, in particular as we already are analysing how frequently the T/E measure leads to a false interpretation of the content.

The relative measures are applicable to the evaluation of any fragment collection but in particular to those where the fragments have been further processed, e.g. by appending some of the fragment context information to the fragment bodies. If these fragment collections are measured with the absolute measures, we could get coverages greater than 100% which is rather useless regarding the goals of the evaluation. Therefore, the relative measures introduce a third dimension to the evaluation: a system for XML retrieval with a set of test queries. Different collections are then compared by the quality of the search results, given that the XML retrieval system and the queries are a constant factor.

[2]The bytesize of the parsed document is a comparable unit whereas the bytesize of the file is not, as it may be different even for equivalent XML documents.

If we have access to the set of relevant answers which is also known as the *ideal recall base*, we can measure the proportion of the relevant answers that is included (or alternatively, not included) in the fragment collection and thus available to the XML search engine. Because of the hierarchical structure of the fragments and because of the possible overlap in the ideal recall base, we may need more complicated ways to measure the proportion than simply counting the relevant fragments in the collection. For example, we can take the near misses into account by considering the parent and child nodes of the relevant elements. Luckily, several evaluation metrics have been implemented for measuring various aspects of Information Retrieval including those mentioned in this section. Such metrics include combinations and variants of the traditional metrics such as precision and recall.

4.5.2 Division examples evaluated

The example documents introduced and divided into fragments in Section 4.4.4 are now evaluated as if they were small fragment collections. Evaluating the IR performance of the division of a single document is not meaningful because reliable IR testing requires larger quantities of test material. However, the absolute measures can be evaluated of the divisions of single documents.

When the small article is divided into fragments, the size of the fragment collection comes down to 16,923 characters which is 99.19% of the size of the original article (17,061 characters). The discarded Text nodes contain only 138 characters. However, when the big article is measured after fragment selection, we only have 151,498 out of 182,497 characters (83.01%) included in the fragment collection. The differences in the coverage can be partly explained by the bibliography in the small article (626 characters) that was mistakenly included because its parent element is a qualified full-text fragment, but even if the bibliography is excluded, the coverage would still be as high as 95.52%. These numbers demonstrate how, regardless of the size unit, a 100% coverage is not a property of successful fragment selection.

When measuring the error rate of the algorithm, we must remember that misjudging full-text as data is a more serious error than misjudging data as full-text because a fragment classified as

data will not be indexed as an independent unit. When data content is classified as full-text, e.g. part of the back matter in the small article, we may still recover from the error with good text and language processing techniques. The small article did not have any full-text that was misjudged as data — in fact, no data content at all was detected — whereas only the front matter and the bibliographical entries were considered data in the big article, which results in a 0% error rate. Regarding the other point of view, we can identify three data fragments that qualified as full-text fragments: one list of keywords, one bibliographical entry, and one back matter containing a list of references. Out of the 169 elements that were tested for full-text likelihood, a total of 72 were selected as full-text fragments while only 69 out of the 169 elements could actually be truly considered full-text fragments. The error rate thus equals 1.8% (3/169). The overall performance grade of the algorithm seems good as all of the full-text content in the articles is included in the fragment collection, and moreover, some of the unimportant content is excluded.

4.5.3 Ideal fragment selection

So far, we have discovered what can be measured from a collection of indexed fragments, and we have also seen some numbers concerning the evaluation of the example articles. We will next focus on the grading of the measured values. The starting point to our problem is that if we want to index all full-text without too much data-oriented content, a 100% coverage does not describe an ideal fragment collection. Then we ask, what is the coverage of an ideal fragment collection or do we even have to know it? As expected, single right answers cannot be given to these questions.

The ultimate goal of the evaluation is to find out which values of the absolute measures yield the best IR performance with an arbitrary document collection and what is required of the related IR techniques. However, a sufficient goal for now is to find parameter values or settings that, first of all, do not prevent the system from achieving the ideal IR performance and second, that aid the system in reaching towards the ideal performance. Even this goal may turn out to be utopia according to two simple claims: 1) the fewer fragments we search, the easier finding the relevant ones is, and 2)

the fewer fragments we index, the fewer queries will have access to the ideal set of answers. Anyway, the best performance is expected from fragments that comply with the qualities of an XML Full-Text fragment as defined in Section 3.2.1. These qualities cannot be measured with any simple metric because their evaluation is based on human perception.

Assuming that the ideal fragment collection can be selected from one document collection, the absolute values can be measured and either generalised or regularised in order to be applicable to other collections. However, not all values generalise in all cases. For example, the number of characters is dependent on the character encoding, and the number of words is specific to the language of the documents. Moreover, document collections are different from each other in that the proportion of full-text and data varies from one document and one collection to another. If the properties of a collection can be regularised, we can define a set of rules by which the values of the same properties can be inferred for other collections.

Measuring performance and the overall effects on the quality of Information Retrieval is more complicated because other techniques are always involved, e.g. clustering, query processing, scoring, score combination, relevance feedback, and query expansion. Because division into fragments is one of the first steps of indexing XML, comparing different fragment collections is difficult: the consequent steps cannot always be repeated identically.

Performance at different levels of granularity is especially challenging to compare. Some precision and recall metrics are favourable to exhaustive answers whereas others prioritise specificity of the answers. When the size of a relevant answer plays a role in the evaluation metric, the choice of granularity sets intrinsic boundaries on the achievable scores. If a score combination method is applied so that the granularity of the returned fragments may be decided at the time of query processing, a fixed granularity will not be the problem. However, after score combination, the fragment collection is no longer evaluated on its own merits.

CHAPTER 5

Fragment expansion

When the indexed fragments have been selected but not yet isolated from the document collection, we can apply the techniques of *fragment expansion* to the selected fragments. The purpose of fragment expansion is to help an XML search engine sort the fragments by relevance with respect to any given query. A good technique aids in the task by making the fragments more self-descriptive. Two different approaches are studied: 1) How to find metadata in the contextual information which is available in the document but not in the fragment itself, and 2) how to locate and weight the descriptive content inside the fragment. The XML structure of the fragments plays an essential role in both approaches, which makes fragment expansion the last chance to utilise the fragment context information and the XML markup before the fragments are separated into independent entities and converted into another format, e.g. plain text.

This chapter is organised as follows. The methodology that XML markup supplies to the content provider for adding style and structure to the written text are studied in Section 5.1. Section 5.2 is devoted to analysing the XML structure. Whether unsupervised algorithms for the analysis are obtainable is also investigated. In Section 5.3, we consider the possibilities regarding different weighting schemes in the vector space model.

5.1 Markup semantics

The inspiration for finding meanings in the markup of structured documents comes from similar research on natural language where various interpretations of the semantics of style have been proposed [Lan70]. For example, by analysing the semantic components of the sentential structures we can classify texts into distinct stylistic categories [LS81]. In a similar fashion, we strive to understand how semantics can be encoded in XML documents and how to decode the information written "between the tags". Instead of concentrating on the lexical and grammatical categories of a natural language, we now collect statistics of different types of nodes and node structures of markup languages.

5.1.1 Writer's point of view

In this context, a writer is a person who produces the full-text content for XML documents with a writing tool such as a text editor or a word processing application. Practice has shown that professional writers rarely produce or edit raw XML, but the writing tool converts the documents into an XML format instead. In order to define the expected qualities of the resulting markup, we need to determine whether we can assume a lossless conversion, and if not, to what extent and how the writer's intentions are preserved during conversion. An automatic conversion into a standalone XML document often results in a number of XML attributes describing the style of each XML element, which shows in redundant style definitions and causes overhead in the form of big file sizes. However, XML is making its way to the heart of more and more applications[1] where it will be serving as a native file format for text documents, after which there is no longer need for conversions. In these cases, most of the information about the style and the layout of the document is stored in separate style files, i.e. stylesheets or document templates, which fall outside the scope of this thesis.

There are no official guidelines for writing or producing XML although, for each individual document type, they often are speci-

[1] Sun's StarOffice and the open source project OpenOffice currently support XML. Microsoft is also contemplating on following the trend.

fied. For instance, HTML is a widely used document type with more books, tutorials, and online guides than one can imagine, but the instructions are specific to HTML documents and therefore they are not directly applicable to arbitrary XML documents. Other document types have similar problems, which partly explains why not XHTML nor any other XML document type has become the golden standard for full-text documents.

Furthermore, XML does not come with any rules for defining a document type for a specific purpose. For example, full-text content can be stored in element content just as easily as in an attribute value as far as the XML specification is concerned, although the latter option is unusual and often also unpractical. However, the characteristic features that are specific to XML are available in all XML documents regardless of their document type. We will now look into some commonly used techniques for marking up text in XML documents.

References In order to add credibility to the text, writers may refer to previously written authoritative work. Other motives such as quotations and citations are also common for referring to other texts. The references contain identifiers of the referred content such as a combination of the author's surname and the year of publication. When marked up in an XML document, an identifier can be a URI-reference pointing to a resource in another XML document or the value of an IDREF-type attribute pointing to another element in the same XML document. The identifier is associated with an element which typically occurs at the inline level and contains or trails a few carefully selected words as the anchoring text.

Labels of structure Titles of various sizes and levels make the structure of the text clearer and more intuitive and also easier to read. Titles are usually the only content of certain title elements which commonly occur either in the beginning of the corresponding content element or as its immediate precedent. In these cases, the path in the document tree from any fragment to the nearest preceding title would be "first child or previous sibling of the parent node". A good title is considerably shorter than the content it describes, which also shows in the sizes of the title elements. A challenge is, however, that the element names of the title elements vary from one document type to another, and sometimes even one document type may suggest different element names for titles at

different levels, e.g. article title and section title.

Labels of concepts New concepts and essential phrases are often *italicised* by tagging the phrase with appropriate tags. Other temporary changes in the typeface are used for the same purpose: to make a portion of text stand out from the surrounding content. As a result, the text with a differing typeface becomes the content of an inline-level element. This is common to all document types. Only the name of the inline-level element varies from one document type to another. It is insignificant whether the style is defined in a separate stylesheet or encoded in the tag name because an element is needed in both cases for marking the part of the text which the different style is applied to.

Other labels From the point of view of an XML parser, highly annotated full-text in XML format looks remarkably different from unannotated text. Breaking up the full-text into individual words and phrases and surrounding them with new tags increases drastically the number of text nodes and decreases the average length of the text nodes. However, annotation of text can hardly be expected of content producers. For one thing, it requires extra work from the writer, and for another, no standard conventions exist for annotating textual entities. Nevertheless, automatic annotation algorithms can be successfully applied to the contents of full-text documents [CHS04, DEG+03].

5.1.2 Semantic units of XML

From the writer's point of view, we move on to the XML processor which handles parsed XML documents and passes on the full-text content to appropriate indexing tools. Like an XML parser, a general-purpose XML processor is blind to the semantics of element and attribute names as it does not assume any particular document type. However, we do assume that there is semantic information encoded in XML markup as seen by XML processors.

Analogical reasoning in linguistics starts from the morphemes of natural languages which are defined as the smallest units of language that are capable of carrying semantic meanings. To adapt the definition to XML documents, we are interested in the smallest units of XML markup that carry information useful in XML retrieval — the morphemes of XML. The most common ones are

introduced in this section.

We consider four kinds of nodes in an XML document which is first parsed into a DOM Node:

1. Element nodes. There is a clear distinction between inline elements and other elements as the former do have text node siblings and the latter do not. The content models of both the element itself and its ancestor elements are another distinctive feature that we look for in element nodes.

2. Attribute nodes. Attribute values are as difficult to interpret as attribute names unless the attribute type is known. The most interesting types are ID, IDREF, and IDREFS. If there is an IDREF-type attribute in one element and corresponding ID-type attribute in another element, we can interpret identical attribute values as a link connecting the corresponding two elements.

3. EntityReference Nodes. The semantic meaning of entity references is not always intuitive, but knowing the type of an entity that is being referred helps us determine its importance. For example, references to character entities can be interpreted as a character encoding issue, whereas references to longer string values may function as phrase markers. These two kinds may appear in both text content and attribute values. Unexpanded external entities are more challenging to interpret as compliant XML parsers only report them as Entity Nodes and, as a consequence, they have a rather unpredictable effect on the true size of the ancestor elements.

4. Text Nodes[2]. The full-text content in XML documents is found in the values of text nodes. Different Text nodes are distinguished by the type of their parent nodes, e.g. elements, attributes or entity references[3].

[2]For simplicity, CDATASection Nodes are also treated as Text nodes in this thesis.

[3]Entity references may be present in the DOM tree although they are expanded by validating XML parsers.

Other types of nodes that are not investigated in this thesis, include Document Nodes, ProcessingInstruction Nodes, Comment Nodes, DocumentType Nodes, Entity Nodes, and Notation Nodes. Although they may turn out to be relevant and even useful, they are either too irregular or too uncommon to have significance at the current stage of the research. In the future, as the XML standards and tools develop, we may also consider other types of nodes such as those for storing document fragments.

5.2 Analysis of full-text fragments

In Section 5.1, we looked at how various writing techniques appear in the XML markup. The point of view in this section is the opposite: we want to study how the writing techniques can be recognised by only analysing the XML markup. Keeping in mind the goal of being independent of document types, we estimate how successfully the semantics of document structures can be detected and interpreted without a DTD and, with a DTD, how the validity constraints may help in the process. As far as it concerns fragment expansion, analysing document type definitions themselves is rather trivial and therefore it is excluded as irrelevant. Besides fragment expansion, we find motivation and inspiration for the analysis in related work which is briefly introduced in this section.

5.2.1 Referred and related content

The purpose of regarding content outside a fragment, in particular the fragment context information as defined in Section 3.1.1, is to make the fragment more self-descriptive and independent and, consequently, easier to find. This is achieved by appending related terms and phrases to the fragment. There are two different ways to find useful content descriptors in the fragment context. Some are located by following explicit links (references) pointing out of the fragment and others by backtracking to the beginning of the document and detecting the structural labels, e.g. headers and titles, on the way.

The inspiration for studying links inside a document comes from the work of Maarek et al., according to whom, proximity of text

```
...and for the first time the world learned of the
&ldquo;Ultra Secret&rdquo;<ref rid="BIBA100321"
type="bib">1</ref> and of
&ldquo;Codebreakers.&rdquo;<ref rid="BIBA100322"
type="bib">2</ref> The details of the existence of
large computational devices was delayed ...
   --- fragment boundary ---
<bb
id="BIBA100322"><au><fnm>D.</fnm><snm>Kahn</snm></au>
<ti>The Codebreakers; the story of secret writing,</ti>
<obi>MacMillan, New York,</obi>
<pdt><yr>1967.</yr></pdt>
</bb> ...
```

Figure 5.1: Lexical affinities relating the word "Codebreakers" and a linked bibliographical entry.

and links within a page is a measure of the relevance of one to the other because of their *lexical affinity* [MBK91]. In practice, the text at the other end of the link may be more relevant to the topic of the anchoring text than the text in the following paragraph. Furthermore, Maarek et al. define that two words share a lexical affinity if they are involved in a modifier–modified relation. Figure 5.1 represents a case where the word "Codebreakers" is modified by the content that is found at the other end of the link, which suggests lexical affinities between the modified word and the bibliographical entry (`bb` element).

When the theory of lexical affinities is applied to any kind of full-text with links appearing at the inline level, e.g. hypertext, we have to relax the requirement that lexical affinities cannot relate words belonging to different sentences. As the sentences in the full-text content are not isolated by an XML parser, the sentence structures are, in fact, completely ignored. However, we seem to find sound modifier–modified relations by simply following links that point to other fragments, including data fragments. The commonly occurring modifiers include both block-level content, such as tables,

figures, and corresponding captions, and out-of-line content, such as notes, footnotes, biographies, and bibliographical entries. If the link points to an element inside the fragment, the referred content is part of the fragment, and although we may still find useful lexical affinities, appending the referred content again will not make the fragment more self-descriptive. The bigger the fragments, the more common this scenario is and the more we could benefit from other ways to utilise the discovered lexical affinities. On a further note, analysing links that point to other XML documents falls outside the scope of this thesis.

Reliable indentification of the attribute types and values requires that the definitions be available either in an XML DTD or a schema definition. Without any definition for the document type, finding the best set of candidate ID and IDREF(S) attributes is an NP-hard optimisation problem where a 100% accuracy is beyond reach [BM03].

When the attribute type definitions are known, link recognition is quite straightforward. Links inside one document are unidirectional, pointing from one element to another.[4] The starting point of the link has an attribute of type IDREF or IDREFS, and the element at the end of the link has an ID type attribute with an identical value. Upon fragment expansion, the content of the element with the ID attribute is associated with the fragment that contains the referring element with the IDREF attribute. A slightly different approach was presented by Liu et al. who map a referring element to a referred element [LZC04].

Associating referred content with the fragment is also motivated by the *Relevant Linkage Principle* [BYR02], according to which, links point to relevant resources. Whether this assumption about hyperlinks in hypertext documents holds for the links in XML documents for scientific articles is studied in Chapter 8.

As seen in the example where a big article was divided into fragments (see Section 4.4.4, details in Appendix A), a section (`sec` element) is sometimes considered too big to be indexed as such, and the subsections (`ss1` element) of the section are indexed instead. However, because a section is typically not merely a union of its

[4]However, several links between two XML elements are possible, which in practice enables bidirectional links.

subsections, some essential content such as the section title is at risk to be neglected in the index. Most of the sections of the big article offer us examples of this kind with the exception of the third section where even more content is left out as too small. If most section titles are not included in the indexed fragments, the self-descriptive value of the fragments deteriorates. To what extent it does so is complicated to evaluate because there are several ways to associate the related titles with the indexed fragments. For example, because the title of a section is related content to all of its subsections, it can be associated likewise assuming that the subsections qualify as full-text fragments. Furthermore, the titles of the corresponding article, journal, and even the section in the journal, can also be considered related content to the subsections, but it is well worth questioning whether it makes the fragment more self-descriptive if the title of a journal is associated with a small subsection. A good compromise might be that the tree distance is taken into account when assessing the importance of a title–fragment relation.

More motivation for paying heed to titles comes from related research: Titles have been found valuable as an asset to Machine Translation thanks to their restricted grammar and vocabulary. Nagao et al. found proof for this in the TITRAN system which specialises in the translation of titles of scientific papers from English into Japanese [NT80, NTHK80, NTYK82]. They also discovered the restricted semantics of the titles. More than 98% of the titles were noun phrases, most of which were carefully selected terminological words specific to a particular field with no polysemic ambiguity. To information retrieval systems, these words are ideal content to be indexed, as they have the characteristics of good search terms, as well.

The structural labels that we are interested in include any kind of titles, subtitles, or headings that can be considered useful in describing the related content. Example (5.1) from a real-world document collection shows a typical case where the title of a section is presented in the content of a `st` element.

(5.1)
```
<sec>
  <st>Conclusions</st>
  ...
</sec>
```

However, looking at the element names is not sufficient as Example (5.2) proves. An empty title element seems common in the beginning of the first section as similar examples are easy to find throughout the document collection of scientific articles. In some cases, we find additional information in the attributes, but in most cases, we have no attributes to look into. Fortunately, titles like "Introduction" do not add a great amount descriptive value to any full-text fragment.

(5.2)
```
... <bdy>
<sec type="intro"><st></st>
  <p>The August 1995 Micro Law reported... </p>
  ...
</sec> ...
```

Example (5.3) presents us a case where the position of the most title-like element (`b` element) is less than typical. It is deep inside the `vt` element it describes. However, it is the first element bearing textual content just like in Example (5.1), which gives us hope for finding regularity in the relative position of titles.

(5.3)
```
... <bm>
<vt id="a100471A">
  <fig>
    <art file="a100471A.gif" />
    <fgc>
      <p><b>James Worthy</b> was named professor
         of management...</p><p>... </p>
      <p>...</p>
    </fgc>
  </fig>
</vt></bm>...
```

To our disappointment, examples contradicting the assumptions about the position of titles are found in the same documents where the titles of whole articles are positioned deep inside the front matter element, after groups of header elements with publication data. Although the article titles all have the same element name, locating them automatically is challenging because they are not found in

the very beginning of any element and because the elements nearby have text content of a similar length.

All these examples are extracted from documents with the same document type. Other document types can be expected to show even more diversity in the element structures around different kind of titles. Consequently, all titles cannot be reliably found in a heterogeneous XML collection without or even with human interpretation of element names. As seen in the examples, there are usually one or more element types assigned to different kinds of titles and headings, however, also style elements can make title elements. In addition, the contrary is true. The title elements are commonly used because of their effect on the style, such as font size, even when the element content is not considered a title. The header elements of HTML (`h1`, `h2`, `h3`, etc.) provide us a rich supply of examples, i.e. the intention of having consecutive header elements with lengthy content in each rarely has anything to do with presenting a group of long titles to the reader. The length of the element content, however, has been found useful in title recognition together with its position inside a document fragment [Hei00]. If we also remember that the nearest related title is not always very near and that sometimes a related title cannot be found, we do have ingredients for a set of heuristics that help us locate the related titles.

5.2.2 Changes in the typeface

Systems that index full-text often neglect the information that is implicitly available in the layout and presentation of full-text. They may perform sophisticated linguistic analysis including full grammatical parsing etc. but, at the same time, they are only capable of processing the plain text presentation of the XML documents where all markup and other information about the text formatting has been removed. In order to fix the shortfall, we want to find methods that analyse formatting properties, typefaces in particular, by taking document structures into account.

Motivation for the analysis comes from the rather simple writing technique of adding emphasis to selected parts of text as described in Section 5.1.1. From what we know about labelling concepts with differing typefaces, we infer the following hypothesis:

A temporary change in the typeface implies a phrase boundary.

Assuming that we can detect the temporary changes in the typeface, the hypothesis should turn out to be helpful in *phrase detection* which in turn aids in building separate phrase indices and enables schemes for phrase weighting.

Thanks to the XML markup, we can locate the portions of text content that are supposed to be formatted with a typeface different from the surrounding content. In order to be independent of document types, we intend to ignore what the specific typefaces are called, what they look like, or even how they differ from other typefaces. Therefore, any kind of change in the typeface is potentially interesting, which makes this theory applicable to all documents where typefaces are used for presentational purposes.

It is necessary to read document type definitions if, for example, we want to know what the intended content models of elements are. Without knowing the content model, we cannot confidently determine whether or not text nodes containing only whitespace can be ignored. Zwol et al. presented an approach where they first analyse the DTD in order to find those elements that can have the mixed content model [vZWD05]. The descendant elements in the mixed content — inline-level elements — are then given a multiplicated weight. Although their results showed slight improvement when inline-level elements were given heavier weights, the overall performance of their system was too low to demonstrate the significance of the idea.

DTDs and schemas also tell us which elements can be used for adding emphasis or changing the typeface. However, by only knowing the element name, we cannot know if the change is temporary in the parenting element or if that typeface is dominant in the whole of the parent element. For example, all the full-text of one document can be presented in italics aslong as the DTD only is concerned. Because of the distinction between temporary and permanent changes in the typeface, we cannot rely on any interpretations of element names but we have to look at the relations of element and text nodes instead.

We know that marking a phrase with a typeface that differs from the surrounding text implies that the phrase be content of an inline element such as the `it` element in Example (5.4).

(5.4)
```
...difference in firing modes. Timed (stochastic)
PN's use the <it>strong firing mode</it> in which
a transition is forced to fire immediately after
it is enabled...
```

This is a typical example of the intention to change the typeface temporarily in order to emphasise the important content. There are numerous other cases where elements appear at the inline level without having anything to do with typefaces. We find typical examples in the XHTML specification [W3C02] which allows tables and lists to appear at the inline level. They should be simple to distinguish, though, as their content differs greatly from that of the inline elements that change typefaces. Table and list elements have structured content, i.e. they contain several child elements, whereas inline elements marking phrases typically have only one text node child. Nevertheless, it is intuitively clear that also inline elements with structured content imply phrase boundaries.

Even when an inline element is used for changing the typeface, it might not contain a useful phrase. Examples include elements for subscript, superscript, embedded program code, preformatted text, mathematical formulae, etc. The content of these elements is often treated differently from full-text content at a later point of indexing and not much harm is done if treated as full-text during fragment expansion. For example, program code is mostly ignored merely due to the large amount of special characters and stopwords such as *if* and *then*.

As a contrast to Example (5.4), the inline elements in Example (5.5) and (5.6) do not contain phrases although they mark temporary changes in the typeface. In this kind of cases, the markup is placed in the middle of a word, and the inline elements are not separated by word delimiters from the surrounding content. More signs of deceptive inline elements are seen in the sizes of text nodes. The text length at the inline level is extremely short (1–2 characters) in Example (5.5), whereas in Example (5.6), the inline elements themselves only contain 1–2 characters.

(5.5)
```
<ti>C<scp>OMPARISON OF</scp>
A<scp>RCHITECTURAL</scp> F<scp>EATURES</scp></ti>
```

(5.6)
```
...through SilRi (<it>si</it>mple
```

```
<it>l</it>ogic-based <it>R</it>DF
<it>i</it>nterpreter, ...
```

In the light of these examples, the original hypothesis should be amended with a remark that a phrase boundary can only occur between words. However tempting it might be, the assumption that words are separated with whitespace characters is specific to languages and writing systems. In languages like English, leading and trailing non-whitespace characters provide negative evidence when deciding the usefulness of an inline element, but languages with other kind of writing systems need different criteria for finding word boundaries.

Discovery of useful phrases is the subsequent goal of detecting phrase boundaries. In addition to the hypothesis, we also claim that the typeface does not change within a useful phrase, as shown in Example (5.4). In order to recognise the significant phrase boundaries, we define a *qualified inline element* which contains a phrase that can be considered more important to the fragment than the surrounding content. The phrase is often emphasised with a different typeface when the text content is displayed, but this is more of a consequence than a requirement. An XML element is considered a qualified inline element when it meets the following conditions:

(C1) The text node siblings contain at least n characters after whitespace has been normalised.

(C2) The text node descendants contain at least m characters after normalisation.

(C3) The element has no element node descendants.

(C4) The element content is separated from the text node siblings by word delimiters.

Defining the lower bounds of n and m improves the quality of detected phrases in the qualified inline elements. If the text node siblings contain fewer than n non-whitespace characters, the inline element is not likely to denote a *temporary* change in the typeface. It might not even change the typeface at all. If the normalised string value of the inline element is shorter than m characters, it is likely that the string is too short to be indexed. Setting the minimum

length of the trimmed phrase m reduces the number of qualified inline elements, which in turn reduces processing overhead. If the values of n and m are set properly, the fourth condition may be redundant.

In the third condition, the definition for qualified inline elements differs most from the HTML definition for inline elements in that qualified inline elements may only contain string-valued content, not other inline elements. This definition is also independent of any tag names and document types. Furthermore, a descriptive definition is more useful than a prescriptive one because it reflects the practice of how tag names are actually used instead of simply assuming the intended usage.

Setting a threshold for the maximum size of an inline element is excluded in the definition, which allows for phrases of arbitrary length to qualify. Fortunately, the amount of excessively big inline elements among the qualified ones is minimal — less than 0.01% — in the document collection described in Section 6.1. The ones that occur in the collection contain several sentences of text. The number of lengthy phrases can be reduced by modifying the condition (C3) as follows:

(C3b) The element has only text node descendants.

With the modified condition (C3b), we have the definition of a *simple inline element* where entity references are not allowed. Because the probability of a phrase containing entity references is proportional to the phrase length, the condition reduces most the number of lengthy phrases. Consequently, some useful phrases similar to the contents of the *it* element in Example (5.7) do not qualify as simple inline elements.

(5.7)
```
... The <it>Audio &amp;amp; Video Recorder</it>
    consists of audio-visual equipment such as ...
```

According to the definition, an inline element does not qualify if more than one element is needed for changing the typeface temporarily. Example (5.8) shows two different phrases where two inline-level elements define the changes in the typeface. The typeface of both phrases is actually subject to three different changes: boldface, italics, and capitalisation.

(5.8)
```
...similar in several respects to Bertrand Meyer's
<b><it>BILINEAR</it></b> [<ref>12</ref>],
pp. 141-146 and <b><it>TWO_WAY_LIST</it></b> ...
```

From our experience, the inline level content where two or more changes have been applied to the typeface consists of mostly of single characters, e.g. italicised superscript, and pieces extracted from program code, e.g. names of variables, functions, or even files. Example (5.8) represents the best phrases in the test collection where two inline level elements are used, and even their quality hardly compares with that of the most common case presented in Example (5.4).

Although detecting cases similar to Example (5.8) would not do much harm, we cannot consider it worthwhile because of the questionable quality of the phrases combined with the added complexity of computation. Moreover, the presupposition behind the heuristics (two or more elements changing the typeface) does not apply to all document types although the heuristics themselves are independent of document types. While some DTDs only define elements for marking temporary typefaces, others may have corresponding definitions for attributes, which implies only one element occurring at the inline level.

Hoi et al. rely on the HTML document type [HLX03], but they are unable to account for cases such as Example (5.9) where the style is defined by inline CSS.

(5.9)
```
<p style="background: blue; color: white;">
A new background and font color with inline CSS</p>
```

Because of CSS and other stylesheet languages, knowing the tag names is not sufficient when analysing HTML documents, and this conclusion can be generalised to other document types as well that allow such structural heterogeneity.

5.3 Indexing expanded fragments

The full-text fragments are ready to be indexed after the analytical part of fragment expansion is completed. How the results of the analysis are utilised when building an index for the fragments is

the question for the next stage. Our ultimate goal is naturally an improved quality of information retrieval. We also want the traditional methods for indexing to be applicable although they do not directly support XML as the document format. Such methodology includes the vector space model characterised in Section 5.3.1. How the content can be weighted without changing the conversion process between the XML presentation and the vector presentation of fragments is considered in Section 5.3.2, while in Section 5.3.3, we focus on selecting the set of applied weighting methods.

5.3.1 The vector space model for XML documents

The first ideas about the vector space model were presented by Gerard Salton as early as the mid 60's [Sal64, SL65]. The ideas developed further in the 70's with the contributions of Karen Spärck Jones, who presented the concept of *inverse document frequency* [Jon72], into what is presently known as the original *Vector Space Model (VSM)* [SWY75]. Adaptations of the original VSM are still used by several search engines such as Google™[BP98] and apparently many others [KT00], which gives account of the high potential of the ideas.

XML documents that are processed as a set of XML fragments differ from the traditional documents in that the fragments share content with each other. For example, the content of each section fragment of a book is included in the content of the book because of the nested structure of the fragments. Applying the traditional vector space model to XML documents is thus nontrivial. Although several XML-aware extensions to the VSM have been proposed [CEL+02, GS02], we choose to get around the need for new models by modifying the structural relations instead inside the XML documents. The problem of vectors representing overlapping documents becomes irrelevant by dividing the XML documents into disjoint fragments using the algorithm presented in Section 4.4. The disjoint fragments will be treated as single documents in the traditional sense of the word.

When XML fragments are converted into term vectors or corresponding data structures, each term is represented by a dimension of the vector with a value denoting its relative weight in the fragment. Carmel et al. presented an interesting XML-aware approach

where terms are paired with their path inside the fragment and each pair is assigned a dimension in the vector [CMM+03]. As a consequence, the same term can actually be represented by different dimensions depending on the context where it appears in the fragment. Nevertheless, we choose to model the fragments as bags of words so that the traditional vector space model applies, and the XML-aware versions are left for further research.

In this thesis, the commonly used terms are defined as follows:

Term Frequency (tf) The number of occurrences of a term in a document fragment.

Document Frequency (df) The number of document fragments where the term occurs.

Inverse Document Frequency (idf) The inverse of the df[5].

Completely new definitions are not needed because the document fragments are treated as documents in the traditional sense — as if they were undivided entities.

The term weighting in the vector space model is often based upon a system proportional to a tf·idf function which assigns largest weights to terms appearing frequently in individual documents but rarely in the document collection as a whole. Both tf and idf can be upweighted or downweighted with the logarithm function or additional multipliers if, for example, we want to include probability factors in the function.

5.3.2 Weighting methods

Because traditional methods cater for plain text representation of documents best, we want to be able to remove all XML markup before the traditional weighting schemes are applied. In practice, the XML markup is removed from the fragments after the weighting methods related to fragment expansion have been applied. The result consists of fragment-sized plain text "documents".

[5]The definition for idf was originally introduced by Karen Spärck Jones in 1972 [Jon72].

By default, the weight of a term comes from the number of its occurrences in the indexed document. This principle concerns all the indexed terms. When the goal is a plain text presentation of the fragments, we only have a limited number of ways to affect the default weights of the content inside the fragment. Basically, we can only insert and delete nodes in the subtree corresponding to the fragment. In practice, we manipulate the fragment content in one or more of the following ways in order to apply special weighting schemes:

Reordering of Element and Text nodes. This affects the phrases and consequently the phrase frequencies of the phrases that occur in the word sequence.

Reduplication or deletion of Element and Text nodes. Both reduplication and deletion affect term frequencies, which is useful when applying weighting schemes. Deletion of nodes additionally decreases document frequencies if all of the occurrences of a term are removed. Modifying the node frequencies also implies changes in the absolute order of the nodes.

Insertion of new Element and Text nodes. Term frequencies increase when new content is appended to the fragment. In addition, the document frequency is affected if the new nodes contain new terms to the fragment.

Some consideration is important when adding new content to fragments. If we associate structurally related content, e.g. titles, with each fragment, the effect on document frequencies is multiplied by the number of fragments sharing the title. If important title words are given reduced weights because of their magnified document frequency, we may hardly talk about a successful weighting method. Applying a zero weight — deletion in practice — does not have such dramatic effects on document frequencies unless the same index terms are deleted from multiple fragments, which sounds unusual.

The overall effect of node duplication depends on the indexing methods. If normalised term vectors model the fragment, duplication of an index term does not quite double the weight of the duplicated word, but it also reduces the weights of other terms because the sum of weights is constant. Consequently, the relative

term weight doubles. Anyway, duplication is still a rather coarse weighting method in general.

Going directly from XML fragments to a vector representation would enable us to define non-integer weights for the content of selected nodes without having to resort to an intermediary coding format between XML and term vectors. However, not having any intermediate fragment format requires the parsing and further processing of the fragment content while performing fragment expansion. This shortcut is a clear restriction on the freedom of choice of indexing methods because it requires that the methods support XML. Therefore, we prefer to have an intermediate bag-of-words type format for the fragments.

Individual fragments can also be weighted as a whole entity. For example, we might want to assign a lower weight to a typical back matter fragment than a section fragment. However, we still face a problem with presenting the weighting factor because the reduplication of nodes only scales well to a small proportion of the indexed content. Once we have a more sophisticated way to present the overall weight of a fragment, we may find fragment size and full-text likelihood useful when computing the weighting factors.

5.3.3 Selecting the fragment expansion techniques

After the analysis, different parts of the fragment can be given different weights. Although all of the fragment expansion techniques can be recommended, the best result might not be achieved by choosing all of them. If we go too far, e.g. if practically all the content of most fragments gets a double weight, we return to where we started from.

Increasing the weight of the qualified inline elements follows the principle, according to which the content emphasised in the fragment should be emphasised in the index. The impact is rather local because the content subject to heavier weighting has a non-zero weight by default. Document frequencies of single terms are not affected at all, but useful phrases should be easier to spot. Regarding the other fragment expansion techniques, many content descriptors, such as titles, table and figure captions, and footnotes, may be completely ignored unless the fragments are expanded with both referred and related content. By default, content that does

not belong to any indexed fragment has a zero weight in the index.

When combined, the fragment expansion techniques affect both term and document frequencies in a mutual vector space, which makes the combined effect challenging to predict. For example, the same terms may occur in both titles and referred content, and by appending both to the fragments, the raise in term frequency may be cancelled out if also the corresponding document frequency grows in the same proportion. Moreover, an elevated document frequency lowers the tf×idf weight of the term in all the fragments, too, where the term occurs. If, however, the words appearing in the appended fragment context information also occur in the fragment bodies, the biggest impact hits the term frequencies, and the effect on document frequencies is less drastic. The optimal solution could be a compromise where either the referred content or the related content is appended to the fragment.

Stemming and other linguistic processing bring more factors of uncertainty to the picture, e.g. two slightly different words may be processed as identical words after stemming. However, considering the linguistic issues in more detail is bypassed in this thesis.

CHAPTER 6

Dividing the INEX collection into fragments

The methodology for selecting the indexed fragments is tested on a real collection of XML documents in this chapter. In order to compare methods that are specific to a document type with those that are independent thereof, we study first what is found when the element names and structures are assumed, and second, which fragments are selected by only analysing the structure of the documents. When the collection is divided into fragments, we are interested in which fragments belong to which levels of granularity, and, level-wise, what kind of properties the fragments of each granularity have.

The test collection consisting of documents of a single document type is introduced in Section 6.1. The DTD is available, so we can choose from two approaches: 1) either we rely on the DTD as described in Section 6.2, or 2) we can ignore information specific to the document type, which is demonstrated in Section 6.3. Approaches proposed by fellow researchers are compared to ours in Section 6.4, as far as they concern the division of the collection into fragments or the selection of the indexed nodes.

6.1 Overview of the document collection

The testbed for the experiments in this thesis has been widely used among our fellow researchers practising XML retrieval as it was the

official test collection (v1.4) for the INEX initiatives in 2002–2004. The collection consists of 125 volumes of scientific journals from the IEEE Computer Society[1] publications, so the common domain of the full-texts is Computer Science. The 860 journals included in the collection were published between the years 1995 and 2002.

The 125 volumes were converted from an unspecified format into 125 XML documents which contain 12,107 elements at the article level roughly corresponding to the size of a traditional "document". The number of XML elements equals 8,239,997, the number of Text nodes 9,751,714, and the number of characters in the Text nodes 390,849,563[2] which comes down to 494 megabytes.

The physical structure of the test collection is irrelevant if the documents are processed with XML literate tools that provide an interface to the parsed XML documents. In several cases, however, the tools and methods are not fully XML-enabled, which magnifies the role of the physical structure of the documents. For example, old scholars may prefer adding light XML support to their legacy tools to re-implementing their ideology with actual XML tools. The different approaches have lead to conceptual clashes that can be explained by taking a closer look at the file structure of the test collection: Each scientific article and article level element is stored in a separate file as an external entity — an article file. The corresponding entity declarations and entity references are found in the internal DTD of each XML document — a volume file. The XML documents also refer to an external DTD which is a common subset of the DTDs of the 125 XML documents.

Because of a confusion with vocabulary, the INEX document collection is often misrepresented; see [FGKL02, FL04]. We will now explain what the collection looks like to XML-oriented people in order to have a common terminological basis with the reader.

INEX XML documents. An *XML Document* is the biggest logical unit of XML. It can be stored in several XML files which are part of the physical structure of the document. When parsed into DOM[3] trees, each XML document has exactly one

[1] http://www.computer.org/

[2] Excessive whitespace has been normalised. The collection contains 394,368,715 characters before the normalisation.

[3] http://www.w3.org/DOM/

Document Node in the tree. The DOM trees representing the whole INEX collection total 125 Document Nodes because the collection consists of 125 XML documents. Each XML document contains one volume of an IEEE journal.

INEX articles. The concept of a *document* has changed because of XML. A document is no longer considered the atomic unit of retrieval. However, XML should have no effect on the concept of an *article*. While it is true that there are 12,107 `article` elements in the document collection, the number of articles is smaller. According to the common perception, many article elements do not have article content. Instead, they contain a number of other papers such as errata, lists of reviewers, calls for papers, term indices, or even images without any text paragraphs.

INEX tags. The specification for XML[4] defines three different kind of tags: start tags, end tags, and empty element tags. A DTD does not define any tags, but it does define *element types*. In the XML documents, though, each non-empty element contains two different kinds of tags. Counting the different tags in the collection (361) is different from counting the different element type definitions in the DTD (192), different element types inside the article elements (178), or different element types that are actually present in the whole collection (183). We ignore the XML attributes in the start tags when counting the different tags in the collection.

INEX content models. The content models of the collection are defined in the DTD. Each element type definition consists of the name of the element and the content model that follows the name. The 192 element types each have one content model, but only 65 of those are different. Of the 65 different content models, only 59 appear in the articles of the collection. For example, the content models of element types `journal` and `books` are not allowed in article content, and elements such as `couple`, `line`, and `stanza` are not present in the collection at all.

[4]http://www.w3.org/TR/REC-xml/

As a contrast to traditional test collections for IR systems, the 125 documents of the INEX collection are too big to be treated as documents in the traditional sense of the word. Consequently, the first challenge of the research is to identify what kind of units of XML are indexed and retrieved. In the following sections, this challenge is answered with methods relying on 1) the DTD, 2) the statistical properties of the markup, and 3) the physical structure.

6.2 University of Helsinki at INEX 2003

The INEX team at the University of Helsinki[5] submitted official runs to the INEX initiative for the first time in 2003. The submission consisted of the maximum number of three runs containing answers for the "Content Only" type queries described in Section 7.2.1. In the creation of the three runs, we implemented the fragment index with two different fixed granularities: 1) sections and 2) paragraphs, both of which were based on a manual analysis of the DTD.

The division into section-sized fragments concerned the subtrees of the following XML elements: `sec` (section), `fm` (front matter), `bm` (back matter), `dialog` (interview), `vt` (vitae). In the document tree, all of these elements are close descendants of the `article` element, and none of them have text node children. In a similar fashion, the paragraph-sized elements taken into account in the paragraph-level division were `p, p1, p2` (paragraphs), `ip1, ip2, ip3` (introductory paragraphs), `bq` (blockquote). These elements have text node children, and also, most of the text content of the collection is covered by choosing these elements. The details presented in Table 6.1 show how heterogeneous the chosen fragments were even when regarding a single element type. In particular, the variance in the sizes can be considered a conflict between the intended usage and the actual usage of the element types. The empty elements included in the fragment collections were no single outliers as there were tens of thousands of fragments in $G_{paragraph}$ that contained less than ten characters.

[5] http://www.cs.helsinki.fi/group/inex/

Element	Count	Min. size	Max. size	Avg.size (chars)
$G_{paragraph}$	985,391	0	7,343	345.70
p	762,223	0	7,343	356.61
p1	25,911	1	1,771	94.05
p2	6,936	3	1,125	91.10
ip1	183,643	0	6,116	349.48
ip2	4,493	0	2,074	184.91
ip3	182	1	2,150	191.41
bq	2,003	0	6,127	358.04
$G_{section}$	110,427	0	110,574	3,695.59
sec	69,735	0	100,373	4,820.91
fm	12,107	60	5,542	756.55
bm	10,065	0	110,574	4,992.83
dialog	194	198	31,735	5,527.48
vt	18,326	0	4,943	623.28

Table 6.1: Fragments of EXTIRP in 2003 in the statistics.

Some overlap was caused by nested elements, e.g. those `p` elements that are ancestors of other `p` elements, and the `vt` elements that are always descendants of the `bm` element. A more detailed description of the entire system called EXTIRP was published in the workshop proceedings [DALP04].

6.3 EXTIRP 2004: Towards heterogeneity

The major changes of EXTIRP from 2003 are described in this section as far as they concern the selection of the indexed units which now meet the requirements for qualified full-text fragments as specified in Section 3.4.4. First, we collect statistics from the document collection as if we could not assume anything about the document type. Selected statistical properties of fragments representing different levels of granularity are presented in Section 6.3.1. Second, the statistical information is applied to the selection of fragments,

Class	Count	Mean size	Median size	T/E	C/E
Frag_{L1}	559,983	403.33	307	2.07	78.83
Frag_{L2}	261,914	1,727.69	844	1.45	56.49
Frag_{L3}	106,043	7,217.66	2,749	1.26	47.32
p	762,223	356.61	281	2.26	129.60
ss2	16,288	1,806.20	1,274	1.45	61.23
ss1	61,492	2,645.18	1,859	1.44	60.12
sec	69,735	4,820.91	2,949	1.43	65.32
article	12,107	32,555.31	26,816	1.18	48.00
journal	860	458,568.27	422,040	1.18	47.86
fm	12,107	756.55	578	0.84	23.87
bm	10,065	4,992.83	3,910	0.80	20.23
bibl	8,551	2,529.43	1,853	0.76	10.18
index	117	20,255.30	14,852	0.88	19.84

Table 6.2: Statistics concerning the size and full-text indicators of evidential full-text fragments and element types.

though in a rather heuristic way at this point. The resulting fragment collection is described in Section 6.3.2.

6.3.1 INEX XML documents analysed

The statistical information collected from the INEX document collection is summarised in Table 6.2. Each of the three categories of fragments Frag_{Lx} represents the level Lx of evidential full-text fragments as defined in Section 3.3.1. Statistics for the other shown fragments that are grouped by element type are shown for comparison.

Elements in the first group (p, ss2, ss1, sec, ...) are representative of typical elements with full-text content at different levels of granularity. As expected, the averaged T/E measure shows values that clearly indicate full-text content for these elements. The second group contains elements that typically have data-oriented content such as the `fm` (front matter), `bm`, (back matter) and `bibl`

(bibliography). Now the T/E values are well below the pivot point indicating data content, and also, the C/E measure shows much lower values for the stereotypical data elements than for the typical full-text elements.

Looking more closely at each granularity level of evidential full-text fragments, we present more selections of the collected statistics. The granularity levels are shown category by category in Tables 6.3–6.5 which are sorted by the frequency of the root element of the fragment. The Average size column shows the mean of the number of characters among the elements that fall in the category.

Element	Count	%	Average size	T/E	C/E
p	355,977	63.57	475.0	2.04	86.92
ip1	108,574	19.39	427.0	2.11	75.91
p1	17,469	3.12	110.4	2.08	36.02
fgc	9,220	1.65	177.6	2.18	49.08
*	559,983	100.00	403.3	2.07	78.83

Table 6.3: Category Fragment$_{L1}$ in detail.

Element	Count	%	Average size	T/E	C/E
ss1	45,560	17.40	2,926.5	1.47	56.18
sec	45,157	17.24	5,222.9	1.46	62.34
li	30,959	11.82	243.3	1.63	33.85
item	25,721	9.82	258.1	1.36	41.64
vt	15,438	5.89	631.9	1.00	156.97
ss2	11,876	4.53	2,143.8	1.47	56.24
*	261,914	100.00	1,727.4	1.45	56.49

Table 6.4: Category Fragment$_{L2}$ in detail.

The first category, Fragment$_{L1}$, seems to be the most homogeneous of the three categories. The reason is clear: mixed content is required of the immediate descendant nodes of the root elements of the fragments. However, the more general requirement of full-text content in at least one descendant element does not offer very strong evidence for the whole fragment being full-text. This is best

Element	Count	%	Average size	T/E	C/E
sec	20,736	19.55	9,551.9	1.43	55.47
bdy	10,788	10.17	29,494.0	1.42	63.73
ss1	10,000	9.43	5,387.6	1.44	48.28
fig	8,696	8.20	402.2	1.66	34.46
list	8,154	7.69	906.3	1.13	35.93
bm	7,432	7.01	5,768.4	0.79	21.15
*	106,043	100.00	7,215.6	1.26	47.32

Table 6.5: Category Fragment$_{L3}$ in detail.

demonstrated in Table 6.5 where the T/E measure shows a rather high variance among the fragments of the same category. We observe that bigger fragments are likely to have both data and full-text content and sometimes a greater emphasis falls on data than on full-text. For example, the back matter (`bm`) elements in the category Fragment$_{L3}$ have an average T/E value of 0.79 which is far below the typical values of full-text fragments. Although some data-oriented content does intrude into the set of evidential full-text fragments, the statistics still seem sufficiently reliable to be used in the definitions for statistical full-text fragments. From another point of view, if the selection of indexed fragments is treated as a classification problem, as mentioned at the end of Section 4.4.3, the statistics are actually used for training the classifier. In either case, our goal is to derive the size range of fragments at each level of granularity.

6.3.2 Fragment selection with derived parameters

The definition for statistical full-text fragments in Section 3.3.2 left us with several open questions. When asked what kind of size ranges define a single level of granularity, we find a partial answer in the statistical information after analysing the document collection. It is intuitive to start from the mean or median sizes of a chosen level of evidential full-text fragments, but we still wonder how much

deviation from the average values is significant and whether there are other ways to utilise the statistical information. At the current stage of research, however, it is still premature to try to define precise correlation factors that describe the correlation of the average fragment size with the minimum and maximum sizes of fragments of the same granularity. Instead, we may consider what kind of ranges are worth testing in order to save time in the rather time-consuming task of experimental testing.

We assume that the lower bound of a size range defines the size of a minimal unit that can be retrieved by the system. If this lower bound becomes unnecessarily small, we are not dealing with a serious shortcoming as the small units make bigger units when combined. However, the computation of fragment relevance is generally more precise when the minimal fragments are bigger and fewer. Too big fragments, however, are difficult to divide into smaller units during runtime if they are indexed as independent units and if they are modelled as a bag of words. To conclude, we require that both the lower and upper bound should be small enough in order for the minimal units to qualify. Moreover, although the bounds can hardly be too low, selecting very small fragments to the index would be pushing the XML search engine to its limits.

We start the experiments with three different lower bounds (100, 150, and 200 characters) and five different upper bounds ranging from 6,000 to 20,000 characters. By combining these bounds into size ranges, we have a set of parameters for the fragment selection algorithm presented in Section 4.4. Selected statistics of eight different fragment collections resulting from the size-based division are shown in Table 6.6. For each granularity, the algorithm was run twice: once with the test for full-text likelihood and once without it. The values of the T/E measure are not shown, but in the full-text (Ft) collections (values parenthesised), the average T/E measure seems rather stable with values between 1.46 and 1.48.

Two of these granularities were represented in the two new fragment collections that were used as a basis for our official submissions in 2004: $d_{[150,8K]}$ and $d_{[200,20K]}$. Full-text likelihood was tested in both cases. The most common paths to the root elements of the fragments are shown in Figure 6.1 for the granularity $\{[150, 8K], T/E \geq 1.0\}$, and for the granularity $\{[200, 20K], T/E \geq 1.0\}$ in Figure 6.2. Each path is preceded by its frequency in the

Division	Fragments (Ft)	Avg. size (Ft)	Coverage (Ft)
$d_{[100,6K]}$	357,274 (348,549)	1,079.50 (1,044.10)	98.68 (93.11)
$d_{[100,8K]}$	260,369 (269,303)	1,487.34 (1,358.67)	99.09 (93.62)
$d_{[150,8K]}$	236,630 (233,989)	1,624.10 (1,545.30)	98.33 (91.69)
$d_{[150,10K]}$	184,181 (187,750)	2,097.16 (1,936.22)	98.83 (93.01)
$d_{[200,8K]}$	216,948 (215,313)	1,755.76 (1,664.32)	97.46 (91.68)
$d_{[200,10K]}$	171,420 (173,351)	2,240.40 (2,082.67)	98.26 (92.37)
$d_{[200,12K]}$	140,986 (144,654)	2,736.68 (2,509.59)	98.72 (92.88)
$d_{[200,20K]}$	86,386 (92,299)	4,498.94 (3,975.05)	99.44 (93.87)

Table 6.6: Absolute properties of fragment collections after size-based division.

fragment collection.

The most common root elements are much like those of the evidential full-text fragments: sections (`sec`), subsections (`ss1`), paragraphs (`p`), and introductory paragraphs (`ip1`). However, the frequencies of the most common root elements are not fully comparable with those of the evidential full-text fragments because the algorithm for fragment selection proceeds top-down whereas the evidential full-text fragments are defined in a bottom-up fashion. Consequently, there can be several evidential full-text fragments under a single root element of a fragment selected for indexing.

For comparison, the most common paths of the fragments that are selected at the same levels of granularity without a requirement for the full-text likelihood are shown in Figures 6.3 and 6.4. The most striking difference is seen in the number of back matter elements (`/article/bm`) which goes up to 7,451 from 2,761 (+170%) if the full-text likelihood is not tested and full-text content is not required in the fragment collection $d_{[150,8K]}$. The corresponding numbers in the collections $d_{[200,20K]}$ are 6,842 and 1,814 which result in an increase of 277%. The number of qualified front matter elements (`/article/fm`) also increases substantially if size is the only requirement for the indexed fragments.

Correspondingly, the number of descendant elements in the `bm`

```
47349 /article/bdy/sec/p
46641 /article/bdy/sec
32487 /article/bdy/sec/ss1
14873 /article/bdy/sec/ip1
13235 /article/bdy/sec/ss1/p
 8720 /article/fm
 7352 /article/bm/vt
 4799 /article/bdy/sec/ss1/ip1
 4155 /article/bdy/sec/ss1/ss2
 3320 /article/bdy/sec/tf
 2923 /article/bm/bib/bibl/bb
 2761 /article/bm
 2254 /article/bdy/sec/ss1/ss2/p
 2242 /article/bm/app
  ...
 1566 /article
```

Figure 6.1: Most common paths to the selected full-text fragments with a size in the range of 150–8,000 characters.

```
40267 /article/bdy/sec
 6826 /article/bdy/sec/ss1
 6683 /article/bdy/sec/p
 6120 /article/fm
 5718 /article/bm/vt
 4802 /article
 2717 /article/bdy/sec/ip1
 1814 /article/bm
 1222 /article/bm/ack
 1202 /article/bdy/footnote
 1158 /article/bdy/ack
 1112 /article/bm/vt/p
 1010 /article/bm/app
  822 /article/bm/footnote
  726 /article/bdy
```

Figure 6.2: Most common paths to selected full-text fragments with a size in the range of 200–20,000 characters.

```
47794 /article/bdy/sec/p
46866 /article/bdy/sec
32749 /article/bdy/sec/ss1
14894 /article/bdy/sec/ip1
12860 /article/bdy/sec/ss1/p
 9932 /article/fm
 8318 /article/bm/bib/bibl/bb
 7451 /article/bm
 5085 /article/bm/vt
 4608 /article/bdy/sec/ss1/ip1
 4184 /article/bdy/sec/ss1/ss2
 3561 /article/bdy/footnote
 3319 /article/bdy/sec/tf
```

Figure 6.3: Most common paths to selected fragments with a size in the range of 150–8,000 characters.

```
40404 /article/bdy/sec
 7020 /article/fm
 6842 /article/bm
 6829 /article/bdy/sec/ss1
 6631 /article/bdy/sec/p
 4832 /article
 2889 /article/bdy/footnote
 2687 /article/bdy/sec/ip1
 1290 /article/bdy/ack
  827 /article/bdy/index/index-entry
  716 /article/bdy
  555 /article/bdy/sec/ss1/p
  451 /article/bm/bib/bibl/bb
```

Figure 6.4: Most common paths to selected fragments with a size in the range of 200–20,000 characters.

elements is greater when full-text content is required of the indexed fragments. As most `bm` elements do not qualify because of the data-oriented content, many of their descendants do, such as the vitae elements `vt` and the appendices `app`. The bibliographical entries (`bb` element), however, are far less frequent in the full-text collections, the path `/article/bm/bib/bibl/bb` having only 2,923 occurrences in the full-text collection $d_{[150,8K]}$ compared with 8,318 occurrences in the other $d_{[150,8K]}$ collection. These numbers also show how the algorithm and the T/E measure are not perfect in separating data from full-text because there are hardly thousands of `bb` elements that contain full-text in the collection. However, the 2,923 `bb` elements only represent 1.25% of the 233,989 elements selected for the fragment collection and 1.96% of the total of 149,168 `bb` elements in the whole document collection.

Further observations were made from the collections representing other granularities. When the minimum size is tuned down to 100 characters, the smallest qualifying fragments do not even contain complete sentences. For example, some publication titles in the bibliographies are longer than 100 characters. Although they represent the data content of the collection, they qualify as a full-text fragment because of their T/E value of 1.0. Other similar portions of XML data also qualify if the lower bound for the fragment size is set very low.

Fragment selection was not the only change in EXTIRP 2004. For example, query expansion was given up due to the lack of human resources. A more detailed description of the system was included in the workshop proceedings [Leh05].

6.4 Comparison to other approaches

The physical structure of the document collection has often been used as the basis for dividing the XML documents into fragments. For example, List et al. emulated flat-document retrieval on XML documents by only including article elements in the index [LMdV+04]. The article elements are physically stored in separate XML files as external entities which are brought into the XML documents through entity references, and it is thus natural to assimilate an XML file to a document in its traditional sense. Moreover, there

are many cases where the scanning for smaller fragments starts from the level of article entities instead of starting from the root of the XML documents [SKdR04, Lar04]. The content outside the articles is ignored in these approaches, such as the context information, e.g. headers in the journal where the articles are grouped by topic. Only few participants in INEX'03 chose to ignore the physical file structure and use the logical structure of XML documents as the basis for their approach [PVG04, STW04].

If information that is not included in the XML Information Set [W3C04] is given a crucial role in an XML retrieval system, the flexibility of the system is seriously compromised. For example, we consider a WWW crawler that fetches documents from remote sites for an XML search engine. In order to find well-formed XML, it has to parse the documents, or otherwise, any document containing "tag soup" would do. Therefore, the XML documents to be included in the index should come through an XML parser or an XML server which do not have to provide the indexing module of the search engine with any other information than that included in the XML Information Set. Among others, the physical file structure of an XML document does not have to be reported. In fact, some XML documents might not even exist in files.

The retrieval of XML elements instead of whole documents is essential in most approaches to XML retrieval. Those with a database-related background typically consider any XML element a potential answer to a query [HLR04, SHBM04], whereas others have found it worthwhile to reduce the number of elements that are first indexed and eventually scored, i.e., by defining a set of indexed element types, by indexing only the leaf-level elements, or by ignoring very small elements. The algorithm demonstrated in the previous section is similar to these approaches in that only a fraction of all the possible elements are indexed, and different from them because it relies on the size and full-text likelihood of the indexed elements.

CHAPTER 7

Test methodology

The purpose of testing fragment collections is to learn how different factors affect the quality of retrieval results. The factors of interest include the parameters for dividing documents into fragments, e.g. the bounds for fragment size, as well as fragment expansion techniques and their combinations. In order to measure retrieval quality, we need to create runs that produce answers from the fragment collection for a fixed set of queries. The answer sets are evaluated with a choice of metrics, and the results are normalised for a fair comparison. These test methods and the testing plan are presented in detail in this chapter.

7.1 Fragment collections under testing

Analysing the qualities of the division process at one granularity level requires the testing of several fragment collections. Testing one fragment collection and evaluating the answer sets takes approximately 24–48 hours of processor time on a 3GHz processor depending on the amount of optimisation. Thence, thorough tests are performed on collections representing only two different granularities. In addition, baseline testing is selectively conducted on divisions at several other granularity levels.

All baseline fragment collections are similar: the document collection is filtered for fragments of an accepted size with no restrictions on the amount of data in the fragment content. The biggest qualifying fragments are included in the baseline fragment collec-

Division	Fragment	Size (characters)			Coverage
id	count	min	max	avg	% chars
Base1	86,386	200	20,000	4,499	99.44
Base1Ft	92,299	200	20,000	3,975	93.87
Base4	236,630	150	8,000	1,624	97.45
Base4Ft	233,989	150	8,000	1,545	91.69

Table 7.1: Statistics about the baseline and full-text oriented fragment collections.

tion. This method is based on the algorithm presented in Section 4.4. The tree traversal only differs in that full-text likelihood is not tested, and all the fragments of the right size qualify.

The fragment collections resulting from each baseline division are described in detail in Table 7.1. The baseline division at granularity level X is dubbed BaseX. The size of a fragment is defined as the total number of Characters in the Text node descendants after whitespace characters have been normalised. According to this definition, the coverage of the fragments is calculated by dividing the size of the fragment collection by the total size of the document collection which equals 390,849,563 Characters.

The corresponding full-text oriented fragment collections are also shown in Table 7.1 because the set of fragments they contain is different from those resulting from the baseline divisions. The Ft divisions differ from the baseline divisions in that fragments with too much data are not accepted. If the full-text likelihood of a fragment goes below 1.00 by the T/E measure, it is further divided into smaller fragments as described in Section 4.4. The collections involving fragment expansion techniques are omitted as they are redundant: fragment expansion has no effect on the set of selected fragments.

The first granularity [200, 20K] was chosen for testing because it was the basis of one of the official submissions for the INEX 2004 initiative. After several rounds of testing on other granularities, the second baseline [150, 8K] was topping the performance charts, thus rewarding itself a place among the thoroughly tested granularities.

One of the fragments at Base4 granularity is located in the front matter of a journal which is above the article level in the document collection. It is discarded in the actual tests because of limitations set by the test environment of the INEX initiative.

Each granularity is subject to the same set of tests which measure the effects of selective division and fragment expansion. None of these tests are included in the BaseX divisions. The following factors are inspected one by one:

Selective division For a granularity X, a full-text collection BaseXFt is created with the algorithm presented in Section 4.4 which excludes fragments with too data-oriented content.

Association of referred content Division BaseXLi produces a fragment collection where each referred element is appended at most once to the fragment that contains the reference. References are based on the ID type attribute values in the referred element and corresponding IDREF type attribute values in each selected fragment. Self-references are ignored if the referred element is part of the referring fragment.

Weighting of marked-up phrases Qualified inline-level elements as defined in Section 5.2 are duplicated in division BaseXEm under the conditions that the sibling nodes contain at least five non-whitespace characters and the descendant nodes at least three. Further tests are conducted on simple inline elements in BaseXsEm divisions. Triplication of the inline content is performed in divisions BaseXEm3 and BaseXsEm3.

Titles The descriptive value of titles is tested in division BaseXTi by appending the nearest preceding title element to each fragment assuming that one is found in the same article. Recognition of the title elements is based on the DTD.

Performance of the resulting fragment collections is compared to that of BaseX collections when studying the effect of a single factor. In order to discover the facts of the combined effect, further tests are necessary. The additional test pairs comprise BaseXAll divisions where all the four techniques are deployed and BaseX$\mathrm{T}_1\mathrm{T}_2\mathrm{T}_3$ ($\mathrm{T}_i \in \{Ft, Em, Li, Ti\}$, $1 \leq i \leq 3$) where only three out of four factors are present.

An ideal technique improves the IR performance of the baseline whereas the lack of such technique should result in a decline in performance. When measuring the IR performance with a function $f_p(fragment\_collection)$ where f quantises a metric p, the following inequalities should hold for most granularities:

$$f_p(BaseX) < f_p(BaseXIdeal)$$

and

$$f_p(BaseXAll - Ideal) < f_p(BaseXAll).$$

The function f_p can be replaced with any reliable evaluation metric. By comparing the IR performance of the four fragment collections, we can determine the ideality of the selected techniques.

7.2 Test runs

Each test run can be seen as an independent system that first computes similarities between the test queries and the fragments of a given collection, and then returns the answers it considers the most relevant. In this section, we present the run specifications which define how the indices are created from the fragment collection and how the queries are processed. At the outcome of a run, each query is answered with a ranked list of XML fragments sorted by relevance. The result lists will function as the basis for the evaluation of the test run.

7.2.1 Queries

The set of test queries is based on the 32 "Content-Only" (CO) topics of the INEX 2003 initiative, for which the corresponding relevance assessments are available. The Content-Only type topics are traditional in Information Retrieval. They contain a combination of keywords and keyphrases specifying the topic of the user's information need. Keyphrases consist of at least two keywords. The CO topics contain no structural hints or constraints for the XML structure of the answers, which makes the actual queries inherently independent of any document type. By default, any fragment is a potential answer regardless of the tag names or fragment context.

Only the content of the answers matters when their relevance to the topic is judged. An example of a CO topic is presented in Section 8.6 where also the corresponding results are studied in detail.

Converting the topics into queries is rather straightforward. After the explicitly marked phrases are recognised, the topic is presented as a vector. Only the title and keywords parts of the topic are used.

7.2.2 Baseline process

The baseline runs should be simple in that they introduce as few new factors affecting the results as possible. The number of variables should be minimal in order to make the results easy to interpret and compare. Moreover, reduced optionality helps eliminate possible side-effects.

As a first step, two separate inverted indices are built from the fragment collection to be tested. A *word index* is created after punctuation and stopwords are removed and the remaining words are stemmed with the Porter algorithm [Por80]. The *phrase index* is based on Maximal Frequent Sequences (MFS) [AM99]. Maximal phrases of two or more words are stored in the phrase index if they occur in seven or more fragments. The threshold of seven comes from the computational complexity of the algorithm. Although lower values for the threshold produce more MFSs, the computation itself would take too long to be practical. More details concerning the configuration of the phrase index are included in the PhD thesis of Antoine Doucet [Dou05].

Besides the indexed fragments, also the queries are presented as normalised vectors. As we have two indices, we initially compute two relevance scores based on the cosine product for each fragment. One score indicates the similarity of the query terms to the terms in the fragment, whereas the other indicates phrase similarity. The Retrieval Status Value (RSV) is a combination of these scores so that both scores have an equal weight. After all the fragments are ranked by the RSV, the top 1,500 fragments for each query are included in the result lists which consist of disjoint fragments of a fixed granularity.

7.2.3 Additional options

The baseline process is a heavily compromised version of a full XML retrieval system. One of the biggest compromises is the fixed granularity of the answers. Dynamic granularity of the answers does not require anything more but the combination of the RSV scores of the fragment siblings, after which the fragments can be combined into bigger fragments. For example, instead of returning three relevant subsections, it might be more appropriate to return the entire section. This would also enable us to set the granularity of indexed fragments to represent the absolute minimal size instead of a settling with the compromise of indexing answer-sized fragments.

There is no standard score combination method that could be used in a baseline process. As any such method would have a strong impact on the final results returned to the user, the test runs are expected to be more useful without any score combination. It might not be until the fragment selection algorithm and the fragment expansion techniques have been thoroughly tested, when it is the right time to move on and add score combination methods to the test run specification.

Query expansion with words from the answers — even when using only the most highly ranked answers — improves the overall quality given that the answers at the top ranks are relevant [Voo94]. The sooner the precision of the results drops, the greater is the potential gain that query expansion could offer. At this point, however, we first want to find out how to attain the best possible results without query expansion so that the maximum benefit could be made later.

Phrase detection could also be optimised by tuning the parameters for creating MFSs. Although it would lead us to better evaluation scores, the main purpose of the tests is to find out how the indexed fragments should be selected and whether we can take advantage of the document markup with fragment expansion techniques. Optimisation of the process should thus not change the observations that can be made about the tested subjects. Therefore, we are well off with a less than optimal phrase index.

7.3 Evaluation

The evaluation of the test runs is based on the official INEX 2003 evaluation metrics and the final set of relevance assessments[1]. Additionally, results are reported using other metrics which measure a bit different aspects of XML retrieval. In order to normalise the actual scores, we need to construct sets of ideal results which represent the theoretical maximum performance of a search system. In this section, we go through the different factors that affect the evaluation: the complex concept of relevance in XML retrieval is opened in Section 7.3.1, different ways to quantise the assessments are introduced in Section 7.3.2, the metrics are described in Section 7.3.3, and finally, rankings that are ideal according to the assessments are regarded in Section 7.3.4.

7.3.1 Relevance assessments

For each query, the top 100 answers of each official submission were collected into a pool of results to be assessed. A human assessor gave an assessment of relevance to each pooled answer as well as to each relevant element in the article view of a pooled answer. In the INEX 2003 initiative, relevance had two dimensions: *exhaustiveness*[2] and *specificity*. The exhaustiveness of an answer indicates how completely the answer satisfies the information need of the query, whereas the specificity of the answer describes the proportion of the relevant content to the irrelevant content in the answer. Using the notation (exhaustiveness, specificity), both measures are given an assessment value on the scale 0–3 where 0 stands for *not*, 1 for *marginally*, 2 for *fairly*, and 3 for *highly*. For example, a relevance assessment of (3,2) denotes a highly exhaustive and fairly specific answer. More details about the assessments including a description of the assessment interface are presented in the articles by Piwowarski and Lalmas [PL04a, PL04b].

In many ways, the assessments are not perfect. Relevant answers might not be assessed at all if they were not included in the

[1] Version 2.5 from 7 March 2003.

[2] In 2005, the dimension was called *exhaustivity* as the scale was reduced to 0–2.

top 100 answers of any official submitted result set. Moreover, human assessment is always prone to human mistakes, not to mention semantical conflicts in the ways different assessors perceive topical relevance. Nevertheless, the official set of assessments is considered to be a reliable asset for the evaluation of XML retrieval. Thanks to the diversity of the systems providing results to the pool, we can safely assume that nearly all of the relevant content has been assessed, and that the answers judged relevant by the assessors are representative of all the relevant answers to each query.

7.3.2 Quantisation of the assessments

Each metric requires the quantisation of the two-dimensional assessments into a single value in the range [0,1]. While several quantisation functions are available, only the *strict* and *generalised* quantisation are used in the evaluation of the test runs. The corresponding functions $f_{strict}(e,s)$ and $f_{gen}(e,s)$ [KLdV04] are defined as follows.

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

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

Most other quantisation functions are variants of f_{gen} so that some are specificity-oriented whereas some favour exhaustivity. Past experience has shown that the scores produced by these variants are not significantly different from those of f_{gen}. One slightly different quantisation function is called probability $P(R_e)$ of relevance [PG03] which is defined as follows:

$$P(R_e) = \begin{cases} 1 & \text{if } (e,s) = (3,3) \\ 0.5 & \text{if } (e,s) = (3,2) \\ 0.25 & \text{if } (e,s) = (3,1) \\ 0 & \text{otherwise} \end{cases}$$

Although the probability $P(R_e)$ is stricter than the generalised quantisation, it still takes into account the different degrees of relevance.

7.3.3 Metrics: `inex_eval_ng`, GR, and PRUM

The choice of metrics for the evaluation of XML retrieval is still everything but a trivial question because no single metric has become conventional, yet. Since 2002, new metrics have been proposed one after another as the weak points of the previous metrics have been pointed out. As a result, we have access to implementations of different metrics where different aspects of XML retrieval have been addressed. In order to reduce the effect a single metric may have on the results, it is considered important to evaluate the test runs with several different metrics.

As the test queries are part of the INEX 2003 test suite, it is an obvious choice to evaluate the results with the official implementation of the official INEX 2003 metric `inex_eval_ng` [GKFL03]. In principle, the resulting scores are comparable with those of the official submissions of the participating institutions. The biggest improvement in `inex_eval_ng` from the official INEX 2002 metric is the option of considering overlap of the retrieved answers. When the overlap is considered, systems are rewarded for returning unseen relevant content, which effectively discourages attempts to "milk" the recall base by returning relevant descendant elements of previously seen elements. Besides the quantised relevance, the size of a relevant answer is the other important factor that contributes to the score of a returned element.

As the official submissions in 2003 were evaluated according to both the strict and generalised quantisation of the assessments, results in this thesis are also reported by both quantisations. The option where overlap of the answers is ignored is not reported, however, as we see no potential gain in ignoring non-existing overlap. Were the overlap ignored, the number of relevant answers would be bigger, and a set of disjoint answers would yield a lower precision compared to an evaluation where overlap is considered. The reported scores and graphs for `inex_eval_ng` consist of 1) the average precision over all topics which is computed for each topic from the precision values at 100 recall points distributed among the ranks 1–1,500, and 2) curves showing the relation between precision_o and recall_o[3]. The official evaluation software version 2003.007 will be

[3]The subscript o indicates that the overlap of the relevant answers is taken

used when computing the evaluations with the official metric.

During the recent years, `inex_eval_ng` has been criticised for not being stable in all cases, i.e. when evaluating systems that return overlapping elements [KLdV04]. The criticism is justified. If ancestor nodes are returned before their descendants, only the ancestor nodes, weighted by their assessed score, will be considered in the evaluation where overlap is considered. If the order is reversed, both nodes contribute to the overall score. The test runs evaluated in this thesis, however, only contain disjoint fragments. Consequently, small changes in the fragment ranks will not have a drastic effect on the overall evaluation. Another point of criticism concerns the size of the relevant elements which is overappreciated by `inex_eval_ng`. The problem surfaces when huge marginally relevant elements are valued higher than rather small but highly relevant elements. Again, the significance of this problem to our tests is minimal as our tested fragment collections consist of size-controlled fragments.

As a contrast to `inex_eval_ng`, we choose *Generalised Recall* (GR) to be used as an additional metric which was formerly known as the Expected Ratio of Relevant units (ERR) [PG03]. GR indicates the expectation of the number of relevant units seen in the list of answers. For example, if the GR at 20 is 7.5, by viewing the first twenty answers, the user sees 7.5% of all the relevant XML elements. For an ideal answer set, the GR reaches the value of 100.0 after all the relevant answers have been returned. The quantisation of the assessments is implemented as the probability $P(R_e)$ of relevance.

GR does not favour big answers like `inex_eval_ng` as it ignores the size of the relevant answers. Moreover, it does not favour systems returning overlapping elements because by seeing a relevant element, the user is considered to see the overlapping elements, as well. Briefly said, GR favours the smallest highly exhaustive answers.

User's behaviour is taken into account in the third test metric that is chosen for the evaluation: *Precision and Recall with User Modelling* (PRUM) [PD06]. Basically, it favours systems that re-

into account in the amount of relevant content. For example, 100% recall does not require that overlapping answers be returned.

turn a large number of distinct relevant elements or irrelevant elements near relevant ones. The latter are also called near misses which PRUM values higher than completely irrelevant answers. According the general assumptions about the user models, it is highly likely that users proceed from the near misses towards relevant content with a high probability. The different models for user behaviour implemented into PRUM have different probabilities for how users navigate through the document. We have chosen the model where the hierarchical behaviour is expected of the users because a similar user behaviour is assumed in other INEX metrics, such as those based on the xCG [KLdV04]. The probability of navigating from one element to another element is proportional to the sizes of the elements in the assumed hierarchical user behaviour. Unless otherwise mentioned, the PRUM scores are computed with the strict quantisation of assessments.

The online evaluation tool[4] originally developed by Benjamin Piwowarski with implementations of the additional GR and PRUM metrics was used in the evaluation.

7.3.4 Ideal answer sets

An ideal answer set, as intended in this thesis, can be created for any granularity by sorting the collection of disjoint fragments by relevance, after which the fragments are in the ideal order. The ranking is not ideal in the absolute sense, as better evaluation scores may be achieved by allowing for fragments of various levels of granularity to appear in the same ranking. Nevertheless, the ideally ranked list of fragments represents the best result set that can be returned without aggregating the indexed fragments into bigger answers. Consequently, the best result sets are ideal when comparing the IR performance of fragment collections at a single granularity level because the effects of indexing techniques and similarity computation are eliminated. We can thus study whether we gain anything by discarding data-oriented content in the index, or if we lose some relevant content by requiring full-text content of the indexed fragments. Another subject of interest is to see how close to perfect we can actually get in a realistic setting.

[4]Source code available at http://sourceforge.net/projects/evalj/

For both the baseline and full-text collection of each granularity, an answer set simulating an ideal run is built from the assessments. The two-dimensional relevance assessments are first quantised into a one-dimensional scale using the generalised quantisation function $f_{gen}(e,s)$. Given the fixed set of fragments, those that have a non-irrelevant assessment are then sorted with the quantised assessment value as the primary sort key and the size as the secondary sort key, so that the most relevant and the biggest fragments come first. Last, the top 1,500 fragments for each topic are listed in the ideal answer set, although, for most of the 32 topics, there are fewer than 1,500 disjoint answers assessed as non-irrelevant. After all the relevant answers have been used, the rest of the list is filled with irrelevant answers sorted into a descending order by size.

The resulting ideal answer sets are only ideal with regards to the strict quantisation of `inex_eval_ng` and nearly ideal regarding the generalised quantisation, GR, and PRUM. For example, the ideal ranking for the generalised quantisation of inex_eval_ng requires the sort key to be defined as the product

$$size \times f_{gen}(e, s),$$

whereas the ideal ranking for GR with $P(R_e)$ only requires a small change: when answers where (e,s) = (2,3) are given a smaller quantised value than 0.75, the order is ideal. For the PRUM metric, however, the creation of an ideal answer set would require the computation of quantised values for the near misses, too, which are dependent on the elements within a short navigational distance.

For other metrics than `inex_eval_ng` with strict quantisation, the ideal answer sets represent systems of a superior quality instead of being ones that yield the maximum scores. If we built different ideal runs for each metric, we could not properly compare the results with each other as they would describe different systems. Hence, we settle with only one method for creating a perfect ranking for the indexed fragments.

The need to develop a new method for computing the ideal answer set is a point that can be criticised. Other methods have already been proposed in order to evaluate the behaviour of different evaluation metrics in the works of de Vries[5] and Kazai et al.

[5]http://homepages.cwi.nl/~arjen/INEX/metrics2004.html

[KL05], however, none has become a standard. New metrics proposed each year bring up new assumptions about the user models and user behaviour, which in turn imply new requirements for the ideal system for XML retrieval. Since we are not trying to evaluate the evaluation metrics themselves, but we only evaluate different configurations for an XML retrieval system, we define an ideal system as one that achieves either the best or a reasonably high score with any suitable metric. Moreover, the fragment collections that are ranked in the upcoming tests consist of disjoint fragments, which makes the computation rather simple compared with the earlier methods that also have to sort overlapping fragments into an ideal order. How different metrics deal with overlap is an additional factor that, fortunately, can be ignored in this thesis.

The completeness of the ideal answer sets is directly proportional to that of the assessments, which has been tempting to criticise in the past years [HKW$^+$04]. It is more than possible that there are relevant answers in the document collection that did not, however, make it to the top 100 in the result lists of any participant. These answers may not have assessed relevance values at all, in which case they are considered irrelevant in the evaluation. Nevertheless, if some answer is not included in the top 100 of any of the official 56 submitted answer sets, it is highly likely that the answer actually is irrelevant. Moreover, even if a few relevant answers were missing, the ideal answer sets still give us a perspective into what could theoretically be achieved.

The ideal parameters for selecting the indexed fragments can be estimated by comparing the evaluation results of different ideal runs. The ideal parameters vary from one query to another, however, so that whichever fragment collection is valued best for one query might not be best for another. The evaluations are especially different when, for instance, small answers (10–500 characters) are assessed as highly relevant for one query but completely irrelevant for another. Bigger answers are usually not systematically discriminated by the assessors. The average scores are somewhat more stable, and any general conclusions should be drawn from them in order to avoid tuning the parameters towards specific queries. Another thing to consider is that when working with ideal runs, the conclusions drawn by comparing results in an ideal setting might not generalise into any realistic setting.

7.4 Future work

Several areas of this research would benefit from more precise testing whenever the circumstances become favourable. For example, string-valued XML entities consisting of more than one character provide a potential mine of good phrases. Confirming or contradicting this claim requires initial testing on a collection where replacement texts are commonly stored in entities. The INEX test collection is not well-suited for the purpose, but otherwise, full test suites such as that provided by the INEX initiative are required for any quantitative evaluation of the quality of the phrases.

The next step of studying the indicators of full-text likelihood involves tests with the modified T/E measure that takes into account the effect entity references have. The modified measure is expected to further reduce the amount of data fragments in the index. For example, the only misclassified bibliographical entry in the bigger example article in Section 4.4.4 has a T/E value of 1.0, whereas the modified version $\frac{T-2 \cdot R}{E}$ gives the value of 0.6 to the same fragment, thus correcting the classification into one of the data fragments.

The syntax of the run submissions for the INEX initiatives only allows for single-root fragments to be returned as an answer. The assessment tool and the evaluation tool have corresponding limitations. However, allowing multiple roots in the fragment body would be practical in cases where, for instance, a section as a whole is too big to be included in the fragment collection, and most of its 150 child elements are too small on their own. From the INEX 2003 initiative, this would require modification in the result set format, the assessment tool, and the assessment format. Many problematic issues would also arise. When the relevance of a multiroot fragment is assessed, e.g. one that contains the first 50 paragraphs of a section, it is not yet possible to determine which other root combinations should also be assessed for completeness. The assessment procedure changed in 2005 so that some of the new requirements are already supported. The specificity of answers is computed automatically from what the assessor has marked relevant, which has lead to a shift from the four-step scale (0–3) into a gradual scale of 0–100% specificity. The exhaustivity dimension still needs more work and possible new metrics so that we can estimate the exhaustivity of a multiroot fragment. For example, we may want to

define how many fairly or marginally exhaustive answers make a highly exhaustive answer. Charles Clarke proposed an extension to the syntax of the INEX result set [Cla05] but it was not yet implemented for the INEX 2005 initiative. However, more support for these ideas are expected at INEX 2007 with a paradigm shift towards *passage retrieval* [TG06].

CHAPTER 8

Results and evaluation

The evaluation of the major contributions of this thesis — the fragment selection algorithm and the three fragment expansion techniques — is presented in this chapter. After the tests are run as described in the previous chapter, we can analyse the results in order to get some insight on the impact of the proposed methods. For example, we are not only interested in what kind of tasks benefit most from each technique, but we also want to know whether any method is uncalled for in any situation. In general, the purpose of the tests is to study how each method affects the quality of search results.

The analysis starts from Section 8.1 where baseline fragment collections are compared with each other. The fragment selection algorithm with T/E measure as the full-text indicator is evaluated in Section 8.2. The results concerning the fragment expansion techniques are presented in Sections 8.3–8.5, followed by a case study in Section 8.6 where the effects of fragment expansion are studied at the level of a single query. The effects of the tested methods are compared to each other in Section 8.7. Finally, we expand the analysis to cover other granularities in Section 8.8 in order to increase the statistical significance of the results.

8.1 Baseline performance

Baseline fragment collections together with a baseline process are needed for the evaluation of a baseline performance from which the

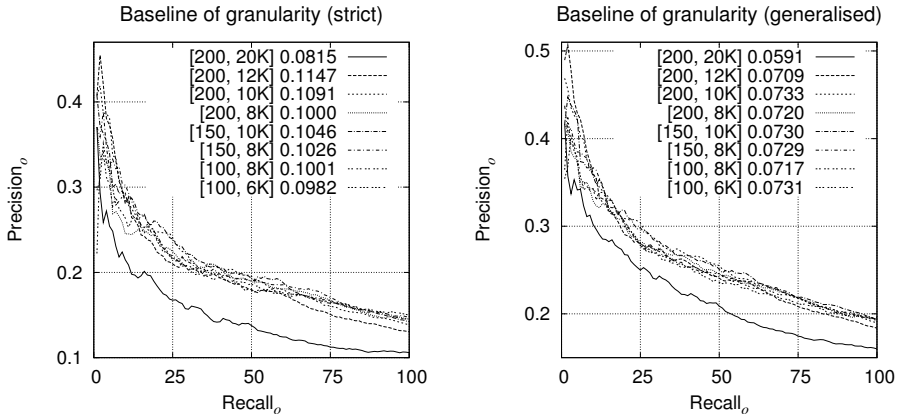

Figure 8.1: Absolute average precisions of eight baseline collections with curves zoomed into the recall levels 1–100/1,500.

relative improvement of the enhanced collections is measured. As we are testing whether fragment expansion improves the potential performance of a fragment collection, we need an individual baseline for each granularity. The performance of the baseline run on baseline fragment collections at a selection of eight levels of granularity is shown in Figure 8.1[1].

The average fragment size which is inversely proportional to the number of fragments in each division seems to be the most significant factor when comparing the average precision at the very low recall levels, e.g. the ranks 1–20 (the first 20 answers for each query). The divisions with the biggest fragments have the steepest curves. Whatever relevant content is found can be included in just a few fragments at the top ranks after which the set of best hits is exhausted and the retrieval precision drops. When the fragments are smaller, returning all the relevant content requires a greater number of fragments, which results in flatter curves.

[1]**How to read the figures:** The quantisation of the assessments — strict or generalised — is parenthesised in the title. Because the evaluated result sets are built from disjoint results, the overlap is considered in the measures of Precision$_o$ and Recall$_o$.

Granularity	strict -o	generalised -o
[200, 20K]	17.00% (0.0815/0.4793)	14.07% (0.0591/0.4201)
[200, 12K]	22.29% (0.1147/0.5145)	15.56% (0.0709/0.4558)
[200, 10K]	20.92% (0.1091/0.5214)	15.72% (0.0733/0.4662)
[200, 8K]	18.67% (0.1000/0.5356)	14.94% (0.0720/0.4818)
[150, 10K]	20.06% (0.1046/0.5215)	15.63% (0.0730/0.4670)
[150, 8K]	19.15% (0.1026/0.5357)	15.09% (0.0729/0.4832)
[100, 8K]	18.68% (0.1001/0.5360)	14.81% (0.0811/0.4841)
[100, 6K]	17.64% (0.0982/0.5566)	14.60% (0.0731/0.5008)

Table 8.1: Baseline precision in proportion with the ideal precision with the absolute precision values parenthesised: baseline/ideal.

The majority of the curves converge at higher recall levels, which by the generalised quantisation shows in rather similar values of average precision after all 1,500 answers have been returned for each query. For the strict quantisation, however, the number of relevant answers is remarkably smaller for each query, and the collections that do well at low recall levels also get best scores overall.

As planned in Section 7.1, the baseline collections of two granularities, [150, 8K] and [200, 20K], will be used as benchmarks when the effects of selective division and fragment expansion techniques are analysed. The baseline precisions for the strict quantisation are then 0.0815 and 0.1026, and for the generalised quantisation 0.0591 and 0.0729.

Besides fragment expansion, which should improve the precision regardless of the granularity of the indexed fragment collection, we are also interested in which size range leads to the best initial precision when answers of a fixed size are returned. In order to fairly compare the baseline runs at different granularity levels, the absolute precision values are normalised by the ideal precision of the corresponding fragment collection. The normalised IR performance of the baseline collections is shown in Table 8.1 with the corresponding ideal precision in the parentheses.

The first obvious question from a reader not yet familiar with

the evaluation of XML retrieval concerns the precision of an ideal answer set: why is it not 1.00 (or 100%)? The answer lies in the *varying recall base* used in `inex_eval_ng`. Relevant elements that are completely included in the previously returned answers are considered irrelevant, thus decreasing the average precision, whereas returning even partially unseen relevant elements increases the precision. The earlier the biggest relevant elements are returned, the sooner is the recall base exhausted regardless of the fact that the big elements may be only marginally specific. Consequently, even if we assume that we have 1,500 elements (or 100%) that are assessed as relevant to some query, our result lists may still contain several elements that do not count as relevant because of total or partial overlap.

Another factor decreasing the ideal precision is that not all the relevant answers are included in the baseline fragment collections as each of them represents a single granularity of fragments. Even when answers of any granularity are available, the ideal ranking of answers when overlap is considered only yields an average precision below 0.68[2]. Nevertheless, when the ideal precision is normalised, it is actually set to 1.00.

The next obvious question concerns the relative precision: how good is a system that only reaches a relative precision of 17–23% while corresponding systems at Text REtrieval Conferences[3] (TREC) achieve remarkably higher precisions? The best systems that participated in the INEX 2003 Initiative achieved relative precisions of 30–40%, but, due to the different nature of retrieved documents, different task definitions, and the comparatively immature evaluation methods, the TREC and INEX results are not directly comparable.

By looking at the average precision values of the collections with ideal rankings, we observe that lowering the minimum fragment size from 200 characters has practically no effect on the amount of highly relevant content in the fragment collection. The highest possible average precision by strict quantisation depends mostly on the maximum fragment size which plays the role of a stopping condition in the fragment selection algorithm. The marginal improvement in average precision by generalised quantisation indicates that only

[2]The perfect run generated by Arjen de Vries, strict quantisation.
[3]http://trec.nist.gov/

the amount of marginally or fairly relevant content increases when fragments smaller than 200 characters are included in the collection. For example, compare the granularities [200, 10K] and [150, 10K]. According to the strict quantisation, the ideal precisions are 0.5214 and 0.5215, whereas the generalised quantisation brings about a small difference in the values of 0.4662 and 0.4670. Similar observations were made by Kamps et al. who tested cut-off values of 20, 40, 50, and 60 words as the minimum size of the indexed elements [KdRS04].

Based on these results, we cannot yet draw confident conclusions about the optimal granularity of the indexed fragments. The reasons are many. Even with a reliable method for *score combination* (1) and the best combination of *fragment expansion techniques* (2), we cannot find the exact size range that can be proven best for the test collection — not to mention any *heterogeneous document collections* (3). The numerous factors of uncertainty include the dependence on the *queries* (4), quality of *relevance assessments* (5), and the simplicity of the *fragment selection algorithm* (6). Nevertheless, the tested baseline performance serves as a good basis for further tests in order to determine a good granularity for the indexed fragments.

8.2 Full-text content required

Fragments that look data-oriented according to the T/E measure are not accepted as individual fragments in a full-text (Ft) fragment collection, which leads to the compared collections containing slightly different sets of fragments despite representing the same granularity. The absolute properties of the tested full-text collections and their counterparts were analysed in Chapter 6. Details about the composition of the tested collections before fragment expansion were shown in Figures 6.1–7.4. For each collection, the average precision measured with four different evaluation methods is presented in Table 8.2.

Comparing the ideal rankings of the different fragment collections is most useful when evaluating what the best composition of a fragment collection would be at a single granularity level. As the ideal rankings are built from the assessments, no operational system

Division	strict -o	generalised -o	GR	PRUM
Base1	0.0815	0.0591	21.0692	1.0231
Base1Ft	0.0923	0.0656	20.7702	1.2621
Base1EmLiTi	0.0949	0.0650	22.0037	1.0914
Base1All	0.0954	0.0686	21.1940	1.2952
Ideal1	0.4793	0.4201	32.8415	6.5695
Ideal1Ft	0.4793	0.4195	31.8924	6.5510
Base4	0.1026	0.0729	27.1790	1.7181
Base4Ft	0.1056	0.0736	26.2124	2.4539
Base4EmLiTi	0.1114	0.0797	27.7506	2.4607
Base4All	0.1170	0.0831	27.6774	2.5398
Ideal4	0.5357	0.4832	46.1536	10.2947
Ideal4Ft	0.5356	0.4817	45.0670	10.7041

Table 8.2: Overall performance of fragment collections with and without data-oriented fragments.

is needed in the evaluation. The unstable factor of computing the relevance scores is thus eliminated. The ideal runs show little difference between the baseline and full-text collections according to the `inex_eval_ng` metric which is known to favour big answers. The fragments of a full-text collection are relatively smaller in size than those of the corresponding baseline collection because the data-oriented fragments, which are not accepted, are further divided into smaller fragments during the selection of full-text fragments.

Accepting only full-text fragments does not increase the amount of relevant content in the fragment collection, and the reason is clear: full-text collections are proper subsets of the corresponding baseline collections. They are also 5–6% smaller in size than the baseline collections. Nevertheless, the amount of highly specific and highly exhaustive content is practically the same in both collections. For example, the number of characters in Base1Ft collection is 5.6% smaller than that in Base1 collection as shown in Table 7.1, but the average precision of the corresponding ideal runs remains the same (0.4793) as far as the strict quantisation is concerned.

Division	strict -o	generalised -o	GR	PRUM
Base1Ft	+13.3	+11.0	–1.4	+23.4
Base1All	+0.5	+5.5	–3.7	+18.7
Ideal1Ft	+0.0	–1.4	–2.9	–0.3
Base4Ft	+2.9	+1.0	–3.6	+42.8
Base4All	+5.0	+4.3	–0.3	+3.2
Ideal4Ft	–0.0	–0.3	–2.4	+4.0

Table 8.3: Percentual change of requiring full-text content (T/E≥1.0) of the indexed fragments.

Regarding the ideal runs, the slightly decreased average precision by the generalised quantisation of `inex_eval_ng` compared with practically unchanged precision by the strict quantisation indicates that the amount of less than highly relevant content decreases by not accepting data-oriented content. The number of fragments that contain this marginally relevant content may, however, increase.

Although the absolute numbers are comparable with the official results, the actual effect of discarding data fragments has to be read between the lines. In order to clarify the capabilities of the T/E measure, the relative effect of requiring full-text content is summarised in Table 8.3 where the full-text oriented collections were compared with their counterparts with no requirement for the T/E values.

Measured with metrics `inex_eval_ng` and PRUM, testing full-text likelihood leads to an improved performance both by itself and with additional fragment expansion. The intuititive observation is that relevant content is easier to find in a fragment collection with a reduced size than in the original baseline collection. Because the amount of improvement is quite different at both tested granularities, further tests on other granularities are required before very specific conclusions can be drawn. However, it seems rather safe to state that requiring full-text content improves the performance of the less-than-ideal operational systems more than it stretches the ideal performance, and as far as our systems are far from perfect,

we benefit from this technique.

By going through the assessments and samples of selected fragments, we find examples that describe the typical effects of testing the full-text likelihood of fragments. There are some cases where the relevant material in the discarded content is isolated from other relevant material, e.g. a small item such a single word or phrase in a bibliographical entry with a relevance assessment of (3,3). Such content is likely to be discarded as data when the indexed full-text fragments are selected and, at the end, it shows in a negative effect on the ideal scores although it might actually be a question of misjudged relevance. There are other cases where the discarded data fragment contains full-text content, such as a back matter element with both a data-oriented bibliography and full-text-oriented appendices. If the potentially relevant appendices qualify, the consequences with respect to the metrics are two-fold. First, the probability of the user seeing the relevant content increases because the content of a relevant appendix is easy to find as a standalone answer or through a direct pointer compared to navigation that starts from the beginning of the back matter. Second, the amount of marginally relevant content decreases as the back matter is not included in its entirety anymore.

To remind the reader, the GR and PRUM metrics favour systems returning many relevant elements (or entry points) regardless of the size of the answers, whereas `inex_eval_ng` favours systems returning lots of relevant content. The difference between the GR and PRUM scores can be explained by the assumption behind PRUM, according to which near misses lead the user towards relevant content and, as such, they should contribute to the overall score. Moreover, the fragments have to be highly exhaustive in order to be considered relevant by GR, and a reduction in the average fragment size leads to a reduced probability of the fragments being highly exhaustive answers to any query. The examples together with the known characteristics of the metrics explain the different evaluation scores, from which we can conclude that although discarding data fragments reduces the total amount of relevant content, the probability of users' seeing it increases nonetheless.

While studying the quality of all the 1,500 answers to each query is useful when a high recall is important, zooming to the beginning of the result list is necessary in order to see how to excel in tasks

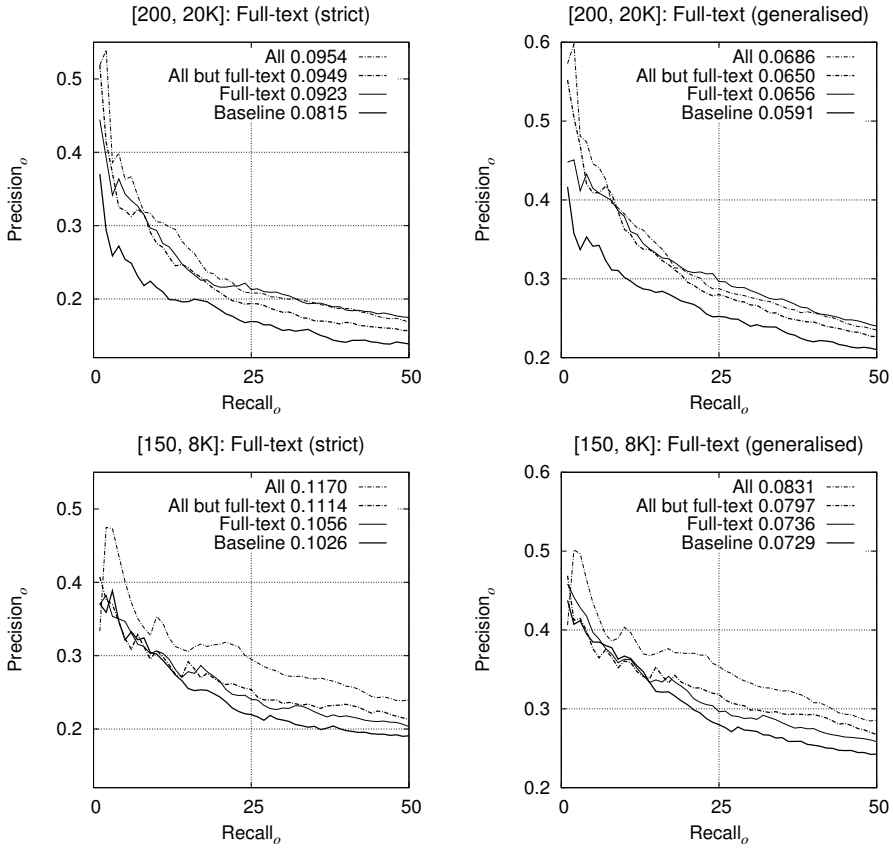

Figure 8.2: Comparable curves showing how the requirement for full-text likelihood (T/E≥1.0) effects the precision at recall levels 1–50.

where high precision is preferred. The curves zoomed where the interesting differences occur are shown in Figure 8.2. What is not shown is that the curves converge towards higher levels of recall.

At the relatively low recall levels of 1–50, discarding data fragments in the index seems to have a strong positive effect when big fragments are allowed. When the maximum fragment size is set down to 8,000 characters, the effect is still positive, but to a lesser extent. The natural conclusion from these observations is

that when we allow fragments up to 20,000 characters in size, we need to require full-text content of them or otherwise some unwanted fragments are indexed. Lowering the maximum size from 20,000 characters down to 8,000 characters also helps discard some of the unwanted data fragments, which shows in a more modest improvement at the granularity level [150,8K].

8.3 Relevant links

The fragment expansion technique where referred content outside the fragment is associated with the indexed fragments concerned 56,161 out of 86,386 fragments (65.0%) at Base1 granularity and 92,502 out of 236,630 fragments (39.1%) at Base4 granularity. The rest of the fragments did not contain references to external resources in other parts of the source document. Although outlinks are unlikely to occur in fragments that consist of whole articles, they are even less common in the Base4 collection that consists of smaller fragments. The measured performance of the fragment collections with linked content and their counterparts without it is presented in Table 8.4.

Division	strict -o	generalised -o	GR	PRUM
Base1	0.0815	0.0591	21.0692	1.0231
Base1Li	0.0966	0.0656	21.9236	2.7439
Base1FtEmTi	0.0980	0.0665	21.5136	1.7760
Base1All	0.0954	0.0686	21.1940	1.2952
Base4	0.1026	0.0729	27.1790	1.7181
Base4Li	0.1054	0.0770	28.1738	2.2771
Base4FtEmTi	0.1072	0.0760	26.7753	2.5364
Base4All	0.1170	0.0831	27.6774	2.5398

Table 8.4: The performance of fragment collections compared with and without referred content.

The differences in the absolute scores measured with different metrics are striking. While GR and `inex_eval_ng` seem to agree about the effect of granularity by giving the best four scores to the collections with smaller fragments, PRUM considers the Base1Li

collection the best. The differences are not surprising if we consider that PRUM neglects the amount of content in a relevant fragment and that it also rewards systems for finding "near misses" where the granularity of the fragment is less than ideal.

When the number of relevant answers is preferred to the quantity of content in the relevant answers, BaseXLi is the best configuration, which is reflected in the top GR and PRUM scores for each granularity. From the fact that `inex_eval_ng` shows less appreciation to the BaseXLi systems, we draw the conclusion that associating the referred content and discarding the other fragment expansion techniques promotes the importance of smaller fragments.

Moving on from the absolute scores, we look at how the evaluation scores are affected when the intra-document links are taken into account. The results which show improvement in most cases are presented in Table 8.5. Appending linked content to fragments seems to have a more positive effect on the bigger fragments at Base1 granularity when other fragment expansion techniques are not used (Base1Li), however, with other techniques (Base1All), the effect is positive only according to the generalised quantisation of `inex_eval_ng`. When the fragments are smaller, this fragment expansion technique improves the score both on its own and with other techniques. When the effect varies a lot between the granularities, we want to be careful not to jump into conclusions too soon, and therefore, the 168.2% improvement is not considered anything but just "improvement" at this point. Therefore, a more detailed analysis is provided in Section 8.5 where we have more data available as the associated titles have an effect on the scores that is very similar to that of the linked content.

Division	strict -o	generalised -o	GR	PRUM
Base1Li	+18.5	+11.0	+4.1	+168.2
Base1All	–2.7	+3.2	–1.5	–27.1
Base4Li	+2.7	+5.6	+3.7	+32.5
Base4All	+9.1	+9.3	+3.4	+0.1

Table 8.5: Percentual change of appending linked content to fragments.

The zoomed curves corresponding to the compared collections

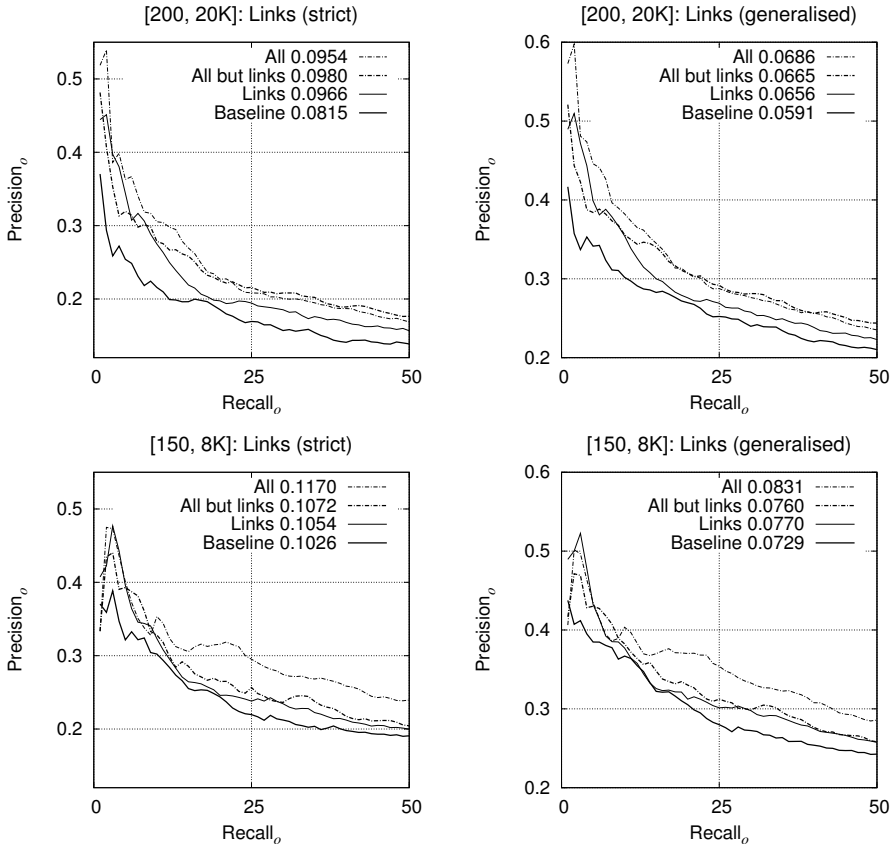

Figure 8.3: The curves concerning the importance of linked content zoomed into recall levels 1–50.

are shown in Figure 8.3. By focusing on the low recall levels, we observe once again that all the baseline curves show the lowest performance, but, contrary to the previous results in this section, the Base1All configuration clearly outperforms the other systems for the same granularity. From this observation, we can naturally conclude that links do point to relevant resources in the test collection and that even with other fragment expansion techniques, the references inside documents should be taken into account in order to maximise the precision at low recall levels.

8.4 Emphasis on the emphasised

In order to picture the magnitude of the potential influence of the structure-based phrase detection and phrase weighting, we present the number of qualified inline elements in Table 8.6. The parameters were set to a minimum of three (3) characters in at least one Text node child and a minimum of five (5) characters in at least one Text node sibling. EntityReference nodes may not appear in the Simple inline elements by definition, but the frequencies show that this additional condition only disqualifies 1.2–1.3% of all the qualified inline elements.

Division	Simple (5,3)	Others (5,3)	All (5,3)
Base1	532,475	6,750	539,225
Base1Ft	523,853	6,743	530,596
Base4	520,965	6,643	527,608
Base4Ft	511,866	6,636	518,502
Whole collection	544,495	7,226	551,721

Table 8.6: Qualified inline elements in the tested collections.

By comparing the numbers of the baseline collections to those of the full-text oriented collections, we can make a somewhat interesting observation: by discarding data-oriented content we also lose many qualified inline elements, 8,622 in the Base1Ft collection and 9,099 in the Base4Ft collection. The amounts are not alarming, though, and we can still consider mixed content a good indicator of full-text content. For one thing, some of the lost inline elements are left out because their source fragment is too small, not because it looks too data-oriented. For another thing, compared to the 5–6% reduction in the total size of the collection, the 1.6–1.8% reduction in the number of qualified inline elements is within reasonable limits.

In order to see real phrases that are emphasised in the INEX document collection, we study Figure 8.4 which shows a selection of qualified inline elements, each preceded by their frequency. The seemingly best phrases are found in those inline elements that have a relatively low frequency in the whole collection. The phrases that benefit most from heavier weighting are such that the words of the

phrase occur in quite many fragments, but they are emphasised in relatively few fragments.

```
67 <it>deterministic</it>
65 <it>weight</it>
65 <it>internal</it>
65 <it>fixed</it>
64 <it>functionally redundant</it>
63 <it>capacity</it>
60 <it>minimum</it>
58 <ref>Lamport's Algorithm</ref>
57 <b>Algorithm</b>
51 <tt>player</tt>
28 <it>sequentially redundant</it>
28 <ref>Sort Partition</ref>
27 <b>primitive</b>
24 <it>dependency relation</it>
21 <it>middle buffering</it>
19 <ref>Hybrid Partition</ref>
17 <b>architecture</b>
17 <it>shape space</it>
16 <it>input buffering</it>
16 <it>false sharing</it>
16 <it>critical path</it>
15 <it>useless shared copies</it>
15 <it>problem complexity</it>
15 <it>perceived usefulness</it>
```

Figure 8.4: Qualified inline elements preceded by the frequency in the document collection.

In order to show when phrase weighting is useful, we consider the phrase "critical path" and the related word frequencies in the 236,630 fragments of the Base4 collection. The word 'critical' occurs in 8,413 fragments, 'path' in 12,927 fragments, both words in 950 fragments, and the phrase "critical path" in 631 fragments. All of these frequencies are directly proportional to the corresponding document frequencies in the tf*idf weighting scheme. As the content of a qualified inline element, the phrase occurs 16 times in 16 different fragments. It is intuitively easy to understand that one phrase is emphasised only once in a fragment although it might occur there several times. The term frequency of the words 'critical' and 'path' thus increases in 16 fragments when the qualified inline elements are given extra weight. As both words have a rather high document frequency, triplication of the inline element might be more effective than duplication. Nevertheless, those fragments

where the phrase is emphasised are expected to be more relevant to corresponding queries than those where the words occur unemphasised.

The actual effect on XML retrieval was studied by creating fragment indices with eight different configurations for both of the tested granularities. Besides the tests where the duplication of inline elements is either on or off, we also tested whether triplication is better than duplication, and whether there is any difference between the simple inline elements and all qualified inline elements. The results of the baseline runs on each index are reported in Table 8.7.

Division	strict -o	generalised -o	GR	PRUM
Base1	0.0815	0.0591	21.0692	1.0231
Base1Em	0.0894	0.0589	21.3262	1.3529
Base1Em3	0.0958	0.0635	21.6545	1.3932
Base1sEm	0.0924	0.0616	21.5971	1.6038
Base1sEm3	0.0886	0.0613	21.6547	1.3999
Base1FtLiTi	0.0969	0.0669	20.9796	1.4314
Base1All	0.0954	0.0686	21.1940	1.2952
Base1sEmAll	0.0918	0.0679	21.0951	1.2591
Base4	0.1026	0.0729	27.1790	1.7181
Base4Em	0.1040	0.0733	27.2178	2.4100
Base4Em3	0.1025	0.0727	27.2384	2.1141
Base4sEm	0.1055	0.0761	26.5913	1.8040
Base4sEm3	0.1029	0.0717	26.3486	2.4839
Base4FtLiTi	0.1133	0.0806	27.6353	2.8075
Base4All	0.1170	0.0831	27.6774	2.5398
Base4sEmAll	0.1138	0.0821	27.1297	2.5799

Table 8.7: Added weight on the emphasised content.

Trying to find some agreement in the results among different metrics is somewhat challenging, so the interpretation of the results may, at times, seem vague. When duplicating the content is the only fragment expansion technique applied, in most cases the definition for the simple inline elements (BaseXsEm) seems to work better than the one that allows for entity references to appear in the

text content (BaseXEm). In the case where triplication is applied, the metrics give a slight favour on all the qualified inline elements (BaseXEm3). With other fragment expansion techniques, the metrics almost unanimously suggest that duplicating only simple inline elements (BaseXsEmAll) is not sufficient: a better precision is achieved by also duplicating inline elements with entity references (BaseXAll).

Whether duplication is more effective than triplication is somewhat dependent on the granularity. In most cases, triplication of the simple inline elements (BaseXsEm3) leads to a lower search quality than the duplication thereof (BaseXsEm). When all qualified inline elements are concerned, however, triplication improves the results more than duplication at Base1 granularity (Base1Em3), but at the granularity of the Base4 collections (Base4Em3), the results are better when the qualified inline elements are only given double weight (Base4Em).

The actual effect of giving heavier weights on emphasised content is shown in Table 8.8 where each collection is compared to their counterpart without additional weighting. The biggest improvement is displayed in the PRUM scores, according to which, however, duplication of inline elements does not improve the results at all when other fragment expansion techniques are applied. Both the improvement and the decline in the results seem to be more pronounced in the Base1 collections where the fragments are fewer and bigger.

In order to find out which weighting scheme works best for the detected phrases, we may try to analyse which configuration leads to the biggest improvement in the evaluation scores. Table 8.8 tells us that, of the two weighting schemes, duplication works better when simple inline elements are weighted at Base4 granularity and when all qualified inline elements are weighted at Base1 granularity, whereas triplication is preferrable when all qualified inline elements of the Base1 collection are weighted. Which weighting method is best for the simple inline elements at Base4 granularity seems to depend on the evaluation metric. We can also draw the conclusion that as the only fragment expansion technique, giving additional weight to the qualified inline elements does not automatically result in a significant improvement in average precision, although, in most of the tested cases, it does.

Division	strict -o	generalised -o	GR	PRUM
Base1Em	+9.7	–0.3	+1.2	+32.2
Base1Em3	+17.5	+7.4	+2.8	+36.2
Base1sEm	+13.4	+4.2	+2.5	+56.8
Base1sEm3	+8.7	+3.7	+2.8	+36.8
Base1All	–1.5	+2.5	+1.0	–9.5
Base1sEmAll	–5.3	+1.5	+0.6	–12.0
Base4Em	+1.4	+0.5	+0.1	+40.3
Base4Em3	–0.1	–0.3	+0.2	+23.0
Base4sEm	+2.8	+4.4	–2.2	+5.0
Base4sEm3	+0.3	–1.6	–3.1	+44.6
Base4All	+3.3	+3.1	+0.2	–9.5
Base4sEmAll	+0.4	+1.9	–1.8	–8.1

Table 8.8: Percentual change of increasing the absolute frequency of qualified inline elements.

The effect of phrase weighting at the low recall levels is pictured in Figure 8.5 which shows the curves associated with all qualified inline elements. From the curves zoomed into the first 50 answers per query, similar observations can be made about the effects of this technique to those that were made of the overall evaluation scores. Although the effect seems more or less positive for the granularity [200, 20K], it seems close to random for the smaller fragments in the Base4 collection. However, all the figures mostly agree that the curves are the furthest apart from their comparative counterpart at the lowest levels of recall, after which they start to converge.

Figure 8.6 shows the average precision at low recall levels with a focus on the simple inline elements. In all the test cases, giving triple weight to simple inline elements leads to a higher average precision than a double weight — but only when the very first answers are considered. Although the improvement is limited to the top two or three answers per query, the observation may be valuable to applications where high precision is preferred. Nonetheless, the triplication of the simple inline elements causes the precision to

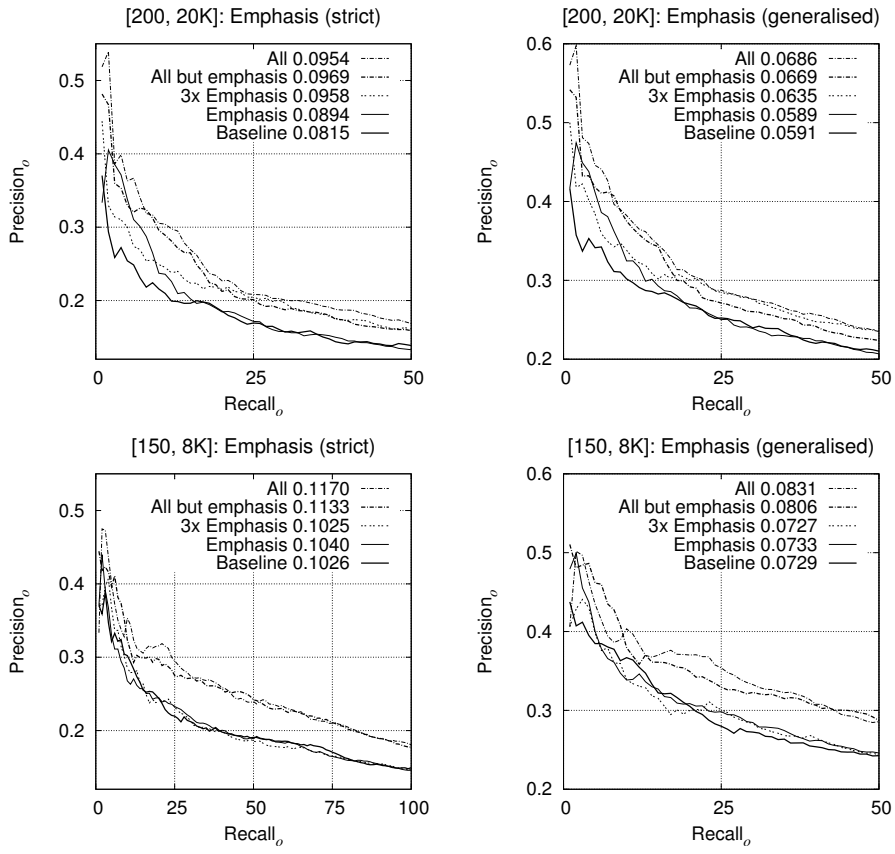

Figure 8.5: Emphasising qualified inline elements has the greatest effect on retrieval quality at the low recall levels of 1–50.

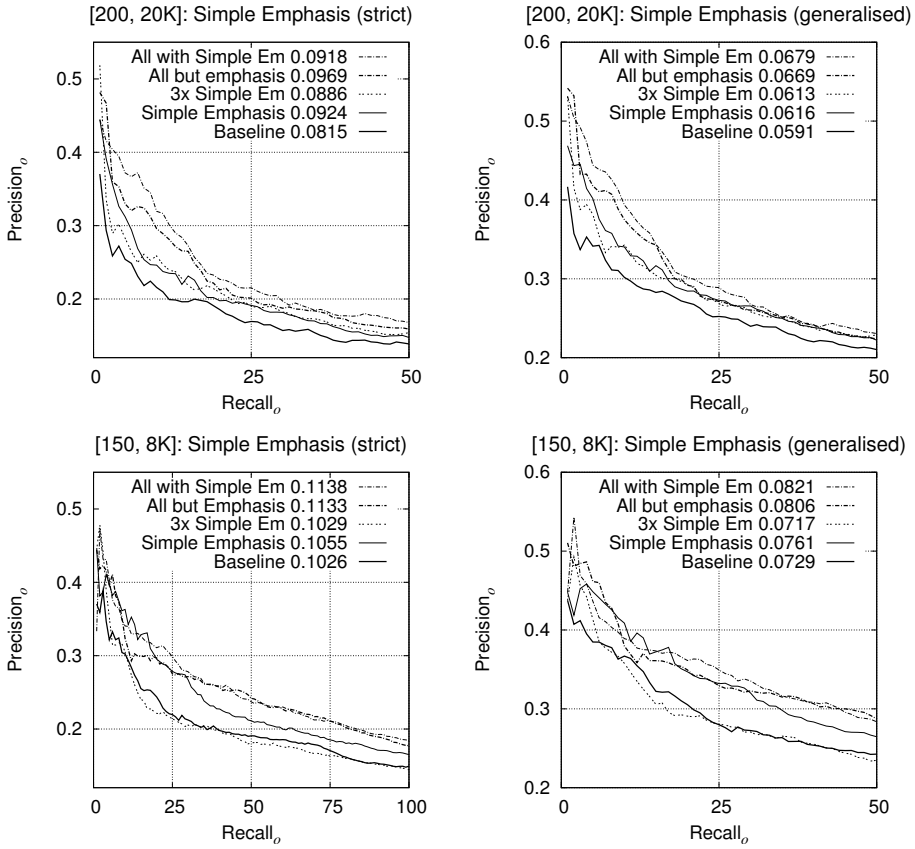

Figure 8.6: The effect of emphasising simple inline elements shown at recall levels 1–50.

sink after the first few answers, and as it sinks even lower than the baseline, it is everything but recommendable for tasks where high recall is appreciated.

Giving extra weight to qualified inline elements has a strong effect on the term frequencies as over 500,000 inline elements are involved. However, the overall effect on an individual fragment depends on the term frequency before the additional weighting. Although duplication of the inline element increases the term frequencies by 1, the actual term frequency rarely doubles because the

same terms tend to occur in the same fragments unemphasised, as well. Further tests with collections of different granularities would produce more data to be analysed, but the most certain conclusion that can be drawn from the results presented is that neither duplication nor triplication alone is a reliable method for improving the retrieval quality. Instead, the variance in the results suggests that we need more sophisticated weighting methods where different inline elements can be weighted individually. For example, we could consider the context of the whole fragment when weighting a detected phrase, so that the corresponding term weights in the whole fragment are doubled instead of duplicating a single occurrence of the terms. Future work on this fragment expansion technique should thus be directed at weighting methods because the quality of the detected phrases can hardly be improved if independence of document types is required of the methods.

8.5 Titles as fragment descriptors

The nearest preceding title was appended to 81,554 out of 86,386 fragments (94.41%) in the fragment collection Base1Ti, and to 235,049 out of 236,630 fragments (99.33%) in the collection Base4Ti. The only fragments with no additional title added were those that comprise a whole article element. The evaluation scores of the fragment collections where titles were added are presented in Table 8.9 with the comparative scores of their counterparts without the additional titles.

By comparing the results at the two granularity levels, we observe quite contradictory scores for the collections where the fragment expansion techniques other than appending the titles are applied (BaseXFtEmLi). According to the strict quantisation of `inex_eval_ng`, this particular configuration is best for the Base1 granularity, whereas for the smaller fragments of Base4 granularity, it is hardly better than the baseline. The generalised quantisation leads to fewer surprises in the relative system rankings: the baseline collections have clearly the lowest scores, whereas the 'All' collections have clearly the highest scores for each granularity.

The relative effect of appending title words to the fragments is shown in Table 8.10. As the only fragment expansion technique,

Division	strict -o	generalised -o	GR	PRUM
Base1	0.0815	0.0591	21.0692	1.0231
Base1Ti	0.0956	0.0631	21.2052	1.2752
Base1FtEmLi	0.0982	0.0665	21.1706	1.3478
Base1All	0.0954	0.0686	21.1940	1.2952
Base4	0.1026	0.0729	27.1790	1.7181
Base4Ti	0.1081	0.0751	27.3364	1.8464
Base4FtEmLi	0.1033	0.0769	27.0489	2.2274
Base4All	0.1170	0.0831	27.6774	2.5398

Table 8.9: The descriptive value of titles measured in absolute evaluation scores.

Division	strict -o	generalised -o	GR	PRUM
Base1Ti	+17.3	+6.8	+0.6	+24.6
Base1All	-2.9	+3.2	+0.1	–3.9
Base4Ti	+5.4	+3.0	+0.6	+7.5
Base4All	+13.3	+8.1	+2.3	+14.0

Table 8.10: Relative improvement of associating titles with the indexed fragments.

including the contents of the title elements in the fragments of the Base1 collection seems to have a strong positive effect on the average precision whereas the positive effect is more modest on the Base4 collection. Together with other fragment expansion techniques, the effect of the granularity is reversed: the other techniques seem to make the titles unnecessary when the fragments are big (Base1All). There is a strong agreement on this behaviour among the metrics `inex_eval_ng` and PRUM with the strict quantisation, which is reasoned as follows.

First, as shown in Figure 6.4, more than half of the fragments in the Base1 granularity are section and front matter elements where the nearest preceding title is the title of the article. Second, the most relevant section elements at the Base1 granularity are more

likely to contain the title words than the smaller fragments at Base4 granularity. Given that, appending the titles does not add any new terms to the most relevant sections but it merely increases the corresponding term frequencies. Third, as sibling elements are considered equal, the words in the article title are also added to the less relevant sections, though, as new terms this time, which causes the df values of the title words to increase and the corresponding term weights throughout the collection to decrease. Fortunately, the less relevant sections are nonetheless relevant according to the generalised quantisation, which explains the 3.2% improvement originating in the titles in the Base1All collection. Fourth, creating associations to the referred content also adds article titles to the fragments, in particular when the references point to bibliographical entries. The proliferation of the title words makes them look more common than they actually are, which in turn causes their descriptive value to deteriorate.

Why these problems do not have a negative effect on the results at the Base4 granularity can be explained by looking into the composition of the fragment index. The fragments of the Base4 collection (Figure 6.3) are considerably smaller than the maximum size of 20,000 characters of the Base1 granularity, the most common fragment root element being `p` (paragraph). The associated titles are now found in the beginning of the sections and subsections, and as they describe smaller portions of text, they are often more specific than the article titles which have to be general enough to label the whole article. Moreover, linked content is associated with less than 40% of the fragments at the Base4 granularity, whereas the corresponding figure was as high as 65% for the Base1 granularity. We can thus come to the conclusion that when the fragments are smaller, e.g. 8,000 characters or less in size, we are less likely to overdo the fragment expansion by associating most useful search terms with too many fragments and inflating their value.

The connection between titles and the linked content is obvious when we compare the percentages in Table 8.10 to those of links presented in Table 8.5 in Section 8.3. While it is clear that the two fragment expansion techniques interfere with each other, we cannot pick out either one of them as the scapegoat. The evaluation scores only show that the best configuration for fragment expansion is highly dependent on the fragment size and the different aspects of

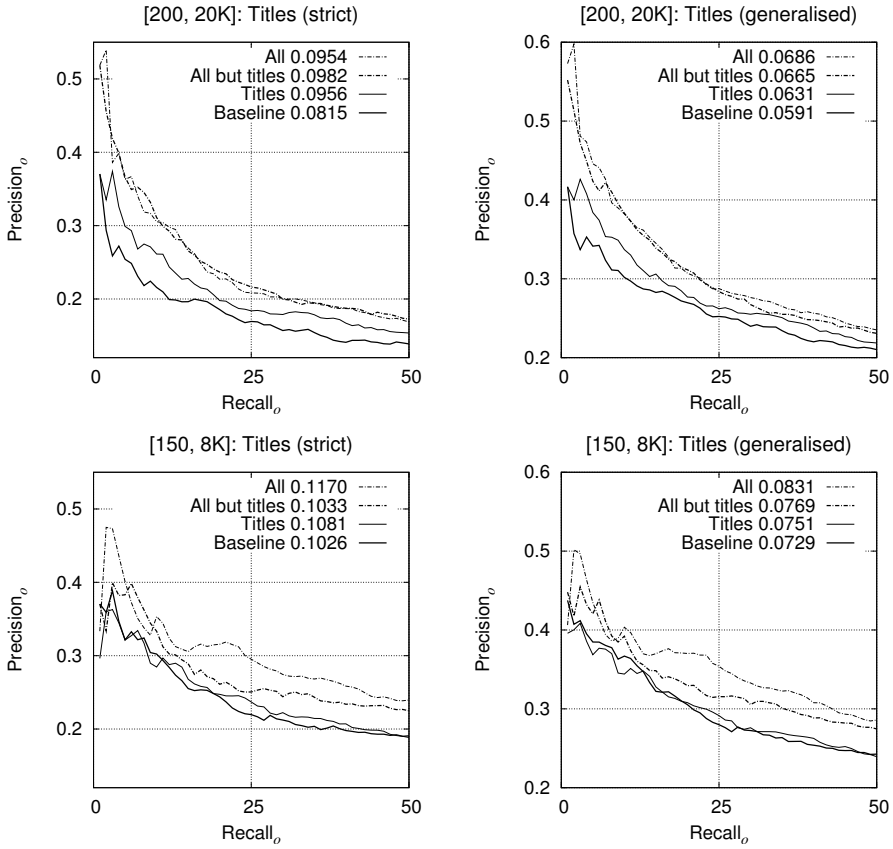

Figure 8.7: Curves demonstrating the effect of associating fragments with related titles.

the retrieval task which are reflected in the evaluation metrics.

The precision at low recall levels is shown in Figure 8.7. By comparing the curves to the corresponding values of average precision in the legend, we observe that they do not seem like a good match at all. What is not shown in the zoomed figures is that the appended titles affect the curves all the way to the 1,500th result for each query, thus improving the recall more than the other fragment expansion techniques. Why the improvement on precision is more modest at low recall levels may be because the importance of the

```
<inex_topic topic_id="124" query_type="CO" ct_no="125">
<title>application, algorithm, +clustering, +k-means, +c-means,
"vector quantization", "speech compression",
"image compression", "video compression" </title>
<description>find elements about clustering procedures,
particularly k-means, aka c-means, aka Generalized Lloyd
algorithm, aka LBG, and their application to image speech
and video compression. </description>
<narrative>interested only in application, and or the algorithm or
procedure applied to speech video or image compression. </narrative>
<keywords>vector, quantization, VQ, LVQ, "Generalized Lloyd",
GLA, LBG, "cluster analysis", clustering, "image compression",
"video compression", "speech compression"</keywords>
</inex_topic>
```

Figure 8.8: The official Content Only topic #124 of the INEX 2003 initiative.

title words is diluted by artificially increasing their document frequencies. Statistical document frequencies should be a good remedy for this as it also makes the methods scalable to the incremental indexing of an infinite number of documents.

Associating related titles with the indexed fragments is the most influential fragment expansion technique presented in this thesis as it affects the term frequencies of nearly all the indexed fragments and the document frequencies of the most descriptive index terms. As a technique complementing the others, it is especially important to the relatively small fragments. The presented results imply high prospects for future research on heuristics for finding titles in heterogeneous XML documents without assuming the names of the title elements.

8.6 Case study: Topic 124

Topic 124 of the official test topics of the INEX 2003 initiative was chosen to demonstrate how essential for accurate retrieval the fragment expansion techniques can be at their best. Figure 8.8 shows the original formulation of the topic. The `title` and `keywords` elements were taken into account when the query was processed

into the presentation of a normalised query vector. The `narrative` element is only meant to aid the human assessor to judge the relevance of the answers, so it is not used when computing the vector products. The `description` element was ignored altogether.

Although only one granularity is represented in this case study, the results should generalise to fragment collections of other granularities, as well, given the particular topic. The configurations of the tested fragment collections together with the results are shown in Table 8.11 which also includes statistics concerning two common terms: 'k-means' and 'clustering'. The different systems in this comparison consist of those already analysed in the previous sections, however, in the first column, they are introduced without the prefix 'Base1'.

System	'k-means'	'clustering'	strict -o	Effect	Rank
Base1	91	1,996	0.0569	—	22
Em	91(4)	1,996(79)	0.1434	+152.2%	8
Em3	91(4)	1,996(79)	0.0808	+42.0%	18
sEm	91(4)	1,996(79)	0.0613	+7.7%	21
sEm3	91(4)	1,996(79)	0.0747	+31.3%	19
Li	104	2,093	0.1572	+176.4%	8
Ti	93	2,098	0.0801	+40.8%	18
Ft	79	1,619	0.0524	-7.9%	24
EmLiTi	106(4)	2,226(79)	0.1634	+187.2%	8
All	93(4)	2,024(79)	0.1997	+251.1%	1

Table 8.11: The number of fragments containing the keywords and the overall effect of fragment expansion at the granularity level of [200,20K].

The second column shows the number of fragments in each collection that contains the term 'k-means' followed by the number of fragments in parentheses where the corresponding inline element was duplicated. For example, in the last configuration 'All', the word 'k-means' occurred in 93 different fragments, and in four different fragments, it was duplicated as content of an inline element. Besides duplication, we also tested the triplication of the qualified inline elements but the results were not encouraging. The best performance was achieved by increasing the frequency of the detected

phrases by one.

In the third column, we present similar statistics for the word 'clustering' which occurs in 1,996 out of 86,386 fragments (2.3%) in the baseline collection. Other search words and phrases are less common in the fragment collection with only one qualified inline element that contains the term 'LVQ' and a handful of those where the terms "video compression" and "image compression" are duplicated.

According to `inex_eval_ng` with the strict quantisation, the average precision with the baseline collection is relatively low for this topic (0.0569), but it is still ranked the 22nd of the 56 official submissions. The corresponding average precision of all the test topics is as high as 0.0815. All the three fragment expansion techniques significantly improve the precision, both on their own and combined. The only configuration resulting in a slight decline regarding topic 124 is the full-text oriented collection with no fragment expansion techniques applied. However, the importance of testing full-text likelihood is best realised by comparing the last two configurations. By adding the test for full-text likelihood to the configuration 'EmLiTi', we get a result list that is ranked first of the official submissions with a precision that is 11% higher than that of the best official submission (0.1799). The purpose of comparing the results with the official runs of 2003 is to show that the improvement from the baseline is significant. If the best test run were ranked among the weakest systems, the real value of the methods would remain a question mark despite over 200% improvement.

Whether topic 124 is an exception or a representative example of a typical query is analysed by regarding the individual evaluation scores of all the topics collectively. The topicwise results are summarised for both Base1 and Base4 granularities in Table 8.12. As different topics benefit from different fragment expansion techniques, the best configuration overall is 'All' where the improvements caused by different techniques are combined. At best, the search results improve for 25 out of 32 topics according to the generalised quantisation while only 7 out of 32 topics suffer. According to the strict quantisation, there were up to seven topics that neither benefit nor suffer from fragment expansion, either because answers relevant to the query do not exist (5 out of 7 topics) or because the existing relevant answers were not retrieved with any configuration

for two topics.

System	Base1: strict	Base1: gen	Base4: strict	Base4: gen
Em	13/12	19/13	15/10	14/17
Em3	14/11	16/16	15/10	17/14
sEm	15/10	21/11	14/11	15/16
sEm3	17/8	19/13	16/9	16/15
Links	20/6	22/10	13/12	22/10
Titles	19/6	19/13	18/7	16/15
Ft	19/6	21/11	14/11	17/14
EmLiTi	18/8	21/11	17/8	24/8
All	18/7	25/7	21/4	25/7

Table 8.12: Number of topics that benefit from the configuration compared to the baseline vs. number of topics that suffer according to `inex_eval_ng`.

Analysing the results topicwise shows that whether a certain fragment expansion technique is useful depends on the search terms and search phrases. Although the biggest improvement and the highest precision may be achieved even with a single fragment expansion technique, the highest probability of improvement and the lowest risk of deteriorating the precision is only achieved by including all the three fragment expansion techniques as well as the selective fragmentation algorithm in the system configuration.

8.7 Summary

While Sections 8.2–8.5 were devoted to analysing the results of each method separately, the same results are compared to each other in this section. Because none of the evaluation metrics has an established position as a standard metric, we give each evaluation method one vote as we study which configurations are better than others. The metrics include all the official metrics of the INEX 2003 initiative, `inex_eval_ng` with both the options of "overlap considered" and "overlap ignored". In addition, the results were evaluated with the PRUM metric under two different settings considering the models for user behaviour. In one setting, we assume the "hierarchical"

System	Avg.	Best Rank	System	Avg.	Best Rank
All	4.17	1 (x6)	Ti	8.17	1
FtLiTi	4.33	1 (x2)	Em3	8.67	2 (x2)
Li	5.06	1 (x3)	Ft	8.89	3
FtEmTi	5.33	1	sEm	9.06	3 (x2)
sEmAll	5.33	2 (x8)	sEm3	10.17	3
EmLiTi	5.89	1 (x3)	Em	10.33	7 (x2)
FtEmLi	6.39	1 (x4)	Base	12.67	8

Table 8.13: Relative rankings of 14 different systems averaged over nine different configuations of evaluation metrics at granularity levels Base1 and Base4.

user behaviour, and in the other, the "NULL" behaviour. Both the strict and generalised quantisation function of the assessments were applied to the settings of `inex_eval_ng` and PRUM. The third metric involved is GR as presented in Section 7.3.3.

How the systems with 14 different configurations were ranked among each other is shown in Table 8.13. The averages are calculated from a total of 18 rankings which come from nine different combinations of metrics and quantisation functions at the two granularity levels of [200, 20K] and [150, 8K].

The 'All' runs where all three fragment expansion techniques are applied to the selected full-text fragments are ranked first a total of six times, whereas runs where only two fragment expansion techniques are used (FtLiTi, FtEmTi, FtEmLi) are ranked first a total of seven times. If the full-text likelihood is not tested but all three fragment expansion techniques are applied to fragments selected only by their size (EmLiTi), the runs are ranked best according to three different evaluation methods. We may conclude from these observations that the whole selection of tricks is not necessary in order to achieve the maximal performance. Surprisingly, though, any single one of the four options may be deselected without an undisputable negative effect on the ranking: any of the four configurations with all but one technique is ranked first, given that the evaluation method is chosen appropriately. Nevertheless, it seems

Granularity	strict -o		generalised -o	
[200, 20K]	0.0954	+17.1	0.0686	+16.1
[200, 12K]	0.1143	-0.3	0.0751	+5.9
[200, 10K]	0.1117	+2.4	0.0785	+7.0
[200, 8K]	0.1139	+11.4	0.0795	+10.4
[150, 10K]	0.1154	+10.3	0.0790	+8.2
[150, 8K]	0.1170	+14.0	0.0831	+14.0
[100, 8K]	0.1087	+8.6	0.0811	+13.1
[100, 6K]	0.1099	+11.9	0.0803	+9.8

Table 8.14: The "All" systems compared to the baseline at eight granularity levels.

that the best chances to achieve good results are available if we take the advantage of all the four techniques that were tested. In addition, not including any of the techniques (Base) leads clearly to the poorest results, which confirms the earlier observations about the importance of fragment expansion.

8.8 Other granularities

So far, we have been comparing the results to the baseline of only two levels of granularity. The results have been mostly positive, but there is the chance that the choice of granularity plays a role in the amount of improvement that each tested technique brings about. In order to get more evidence and thus increase the significance of these tests, we want to widen the perspective by evaluating the "All" counterparts of all the eight baseline collections introduced in Section 8.1. The combined effect of discarding data fragments and expanding the fragments (All) is shown in Table 8.14.

With only one exception, the combined effect of the tested methods is positive. The one negative example is most likely not a sign of weakness as the average precision is relatively high (0.1143), and the score of the corresponding baseline system is exceptionally high (0.1147). The absolute scores are not fully comparable, though, and

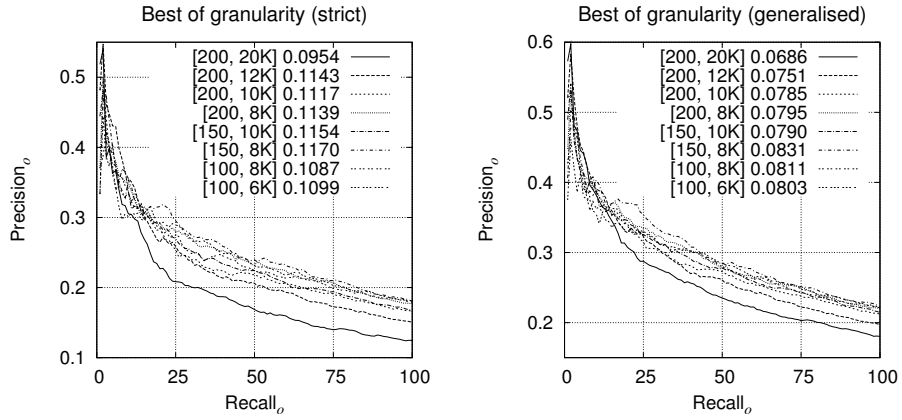

Figure 8.9: Absolute average precision of the "All" configuration of each granularity zoomed into the recall levels 1–100/1,500.

we cannot draw conclusions from these results about which granularity would be the best for the fragment index. The granularity of the retrieved answers in each of the evaluated runs is fixed instead of being sensitive to the query, which is a great difference in nature from operative systems. What we can observe is that all the curves of the runs shown in Figure 8.9 are plotted higher up on the scale than those of the corresponding baselines shown in Figure 8.1.

What was stated about the baseline performance in Section 8.1 also holds for the "All" systems. For example, when the maximum fragment size increases, the precision slightly improves at the first few recall points, whereas, the lowest maximum sizes seem to yield better performance when we go further down on the curve. From the previous sections, we have learned that the positive effect of fragment expansion is expressed the most at the beginning of the result lists, e.g., the first 100 answers out of 1,500. If we restrict our comparison to the first 100 recall points which are shown in the figures, the improvement is clear at all the tested granularities — even at that of [200, 12K] which was the only exception when comparing the average precision over 1,500 answers per query. Based on these observations, it is likely that the fragment selection and expansion methods presented in this thesis improve the quality of

retrieved answers regardless of how the granularity of the indexed fragments is chosen.

CHAPTER 9

Conclusions

In this thesis, we have studied various methods and techniques for exploring and analysing XML documents without knowing anything about the document type. Not being aware of the vocabulary used in element and attribute names, we can only assume that we are analysing well-formed XML, and the range of appropriate tools is quite different from what traditional methods for indexing full-text are based on. The first challenge was to determine the indexed units of text which are usually called documents in the related literature. We call them *qualified full-text fragments* which is a subset of the more general concept of XML fragments. One of the contributions of this thesis was the definition for such fragments which helps us index the full-text content of arbitrary XML documents. Thanks to the indicators of full-text likelihood, we are able to exclude 5–6% of the content from the index without a negative effect on the retrieval quality.

Other major contributions include three techniques for *fragment expansion*. The experimental test results show that this selection of methods improves the overall retrieval precision, and that the effect is emphasised at relatively low levels of recall. In general, the weighting schemes associated with each of the techniques help rank the most obvious relevant answers at the top ranks at the cost of the marginally relevant answers getting a decreased relevance score. This tradeoff is acceptable for tasks where high precision is preferred to high recall. In other words, the XML search applications that benefit most from the proposed methods process queries where relatively few highly relevant answers satisfy the informa-

tion need and where less than highly relevant content is considered irrelevant. Examples of such search environments may also have additional requirements due to low bandwidth, small display, or limited browsing time.

Future work on the topic includes the evaluation of the methods on different document collections in order to confirm the suitability of the methods for heterogeneous XML documents. If the methods turn out to be successful, we will be interested in more sophisticated weighting schemes that would further improve the results. The positive experiences with the INEX test collection as well as future document collections are also likely to encourage us to develop the methodology that is applicable to arbitrary XML documents. For example, we may come up with new or improved fragment expansion techniques, or we may invent something completely different; the chances are unlimited.

References

[ACM+02] Vincent Aguilera, Sophie Cluet, Tova Milo, Pierangelo Veltri, and Dan Vodislav. Views in a large-scale XML repository. *The VLDB Journal*, 11(3):238–255, 2002.

[AJK05] Paavo Arvola, Marko Junkkari, and Jaana Kekäläinen. Generalized contextualization method for XML information retrieval. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 20–27, New York, NY, USA, October 2005. ACM Press.

[AM99] Helena Ahonen-Myka. Finding all frequent maximal sequences in text. In Dunja Mladenic and Marko Grobelnik, editors, *Proceedings of the 16th International Conference on Machine Learning ICML-99 Workshop on Machine Learning in Text Data Analysis*, pages 11–17, Ljubljana, Slovenia, June 1999. J. Stefan Institute.

[AMHHK00] Helena Ahonen-Myka, Barbara Heikkinen, Oskari Heinonen, and Mika Klemettinen. Printing structured text without stylesheets. In *Proceedings of XML Scandinavia 2000*, May 2000.

[AQM+97] Serge Abiteboul, Dallan Quass, Jason McHugh, Jennifer Widom, and Janet L. Wiener. The Lorel query language for semistructured data. *International Journal on Digital Libraries*, 1(1):68–88, 1997.

[AYBS04] Sihem Amer-Yahia, Chavdar Botev, and Jayavel Shanmugasundaram. TeXQuery: a full-text search extension to XQuery. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 583–594, New York, NY, USA, May 2004. ACM Press.

[AYLP04] Sihem Amer-Yahia, Laks V. S. Lakshmanan, and Shashank Pandit. FleXPath: flexible structure and full-text querying for XML. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 83–94, New York, NY, USA, June 2004. ACM Press.

[BCKL02] Daniele Braga, Alessandro Campi, Mika Klemettinen, and Pier Luca Lanzi. Mining Association Rules from XML Data. In *DaWaK 2002: Proceedings of the 4th International Conference on Data Warehousing and Knowledge Discovery*, pages 21–30. Springer-Verlag, September 2002.

[BM03] Denilson Barbosa and Alberto O. Mendelzon. Finding ID Attributes in XML Documents. In *Proceedings of the First International XML Database Symposium (XSym 2003)*, volume 2824 of *Lecture Notes in Computer Science*, pages 180–194. Springer-Verlag, September 2003.

[BP98] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. In *WWW7: Proceedings of the seventh international conference on World Wide Web 7*, pages 107–117, Amsterdam, The Netherlands, April 1998. Elsevier Science Publishers B. V.

[BYFM02a] Ricardo Baeza-Yates, Norbert Fuhr, and Yoelle S. Maarek, editors. *Proceedings of the SIGIR 2002 Workshop on XML and Information Retrieval*, Tampere, Finland, August 2002.

[BYFM02b] Ricardo Baeza-Yates, Norbert Fuhr, and Yoelle S. Maarek. Second edition of the "XML and informa-

tion retrieval" workshop held at SIGIR'2002, Tampere, Finland, 15 Aug 2002. *ACM SIGIR Forum*, 36(2):53–57, 2002.

[BYFSDW00] Ricardo Baeza-Yates, Norbert Fuhr, Ron Sacks-Davis, and Ross Wilkinson, editors. *Proceedings of the SIGIR 2000 Workshop on XML and Information Retrieval*, Athens, Greece, July 2000.

[BYM04] Ricardo Baeza-Yates and Yoelle S. Maarek, editors. *Proceedings of the SIGIR 2004 Workshop on XML and Information Retrieval*, Sheffield, England, July 2004.

[BYR02] Ziv Bar-Yossef and Sridhar Rajagopalan. Template detection via data mining and its applications. In *WWW '02: Proceedings of the eleventh international conference on World Wide Web*, pages 580–591. ACM Press, 7–11 May 2002.

[CDIW05] Jim Challenger, Paul Dantzig, Arun Iyengar, and Karen Witting. A fragment-based approach for efficiently creating dynamic web content. *ACM Transactions on Internet Technology*, 5(2):359–389, 2005.

[CEL+02] David Carmel, Nadav Efraty, Gad M. Landau, Yoëlle S. Maarek, and Yosi Mass. An Extension of the Vector Space Model for Querying XML Documents via XML Fragments. In Baeza-Yates et al. [BYFM02a], pages 14–25.

[Cha00] Hans Chalupsky. Ontomorph: A translation system for symbolic knowledge. In *Proceedings of the 7th International Conference on Principles of Knowledge Representation and Reasoning (KR 2000)*, pages 471–482, San Fransisco, California, USA, April 2000. Morgan Kaufmann.

[Cha01] Soumen Chakrabarti. Integrating the document object model with hyperlinks for enhanced topic distillation and information extraction. In *WWW '01: Proceedings of the tenth international conference on*

World Wide Web, pages 211–220. ACM Press, May 2001.

[Cho02] Byron Choi. What are *real* DTDs like. In *Proceedings of Fifth International Workshop on the Web and Databases (WebDB 2002)*, pages 43–48, June 2002.

[CHS04] Philipp Cimiano, Siegfried Handschuh, and Steffen Staab. Towards the self-annotating web. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 462–471, New York, NY, USA, May 2004. ACM Press.

[CHWM04] Deng Cai, Xiaofei He, Ji-Rong Wen, and Wei-Ying Ma. Block-level link analysis. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 440–447, New York, NY, USA, July 2004. ACM Press.

[CJT01] Soumen Chakrabarti, Mukul Joshi, and Vivek Tawde. Enhanced topic distillation using text, markup tags, and hyperlinks. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 208–216, New York, NY, USA, September 2001. ACM Press.

[CK01] Taurai Tapiwa Chinenyanga and Nicholas Kushmerick. Expressive retrieval from xml documents. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 163–171. ACM Press, September 2001.

[Cla05] Charles L. A. Clarke. Range results in XML retrieval. In Trotman et al. [TLF05], pages 4–5.

[CLO03] Qun Chen, Andrew Lim, and Kian Win Ong. D(k)-index: an adaptive structural summary for graph-structured data. In *SIGMOD '03: Proceedings of*

the 2003 ACM SIGMOD international conference on Management of data, pages 134–144, New York, NY, USA, June 2003. ACM Press.

[CMM+03] David Carmel, Yoëlle S. Maarek, Matan Mandelbrod, Yosi Mass, and Aya Soffer. Searching XML documents via XML fragments. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 151–158. ACM Press, July 2003.

[CMS00] David Carmel, Yoelle S. Maarek, and Aya Soffer. XML and information retrieval: a SIGIR 2000 workshop. *ACM SIGIR Forum*, 34(1):31–36, 2000.

[CMS02] Chin-Wan Chung, Jun-Ki Min, and Kyuseok Shim. Apex: an adaptive path index for xml data. In *SIGMOD '02: Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 121–132, New York, NY, USA, June 2002. ACM Press.

[CMZ03] Yu Chen, Wei-Ying Ma, and Hong-Jiang Zhang. Detecting web page structure for adaptive viewing on small form factor devices. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 225–233, New York, NY, USA, May 2003. ACM Press.

[Col97] Robert M. Colomb. Impact of semantic heterogeneity on federating databases. *The Computer Journal*, 40(5):235–244, 1997.

[CRF00] Donald D. Chamberlin, Jonathan Robie, and Daniela Florescu. Quilt: An XML query language for heterogeneous data sources. In *Selected papers from the Third International Workshop WebDB 2000 on The World Wide Web and Databases*, volume 1997 of *Lecture Notes in Computer Science*, pages 1–25. Springer-Verlag, May 2000.

[CSF+01] Brian Cooper, Neal Sample, Michael J. Franklin, Gísli R. Hjaltason, and Moshe Shadmon. A fast index for semistructured data. In *VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases*, pages 341–350, San Francisco, CA, USA, September 2001. Morgan Kaufmann Publishers Inc.

[Cur97] James E. Curtis. Managing hardcopy documentation in a multiplatform environment. In *SIGDOC '97: Proceedings of the 15th annual international conference on Computer documentation*, pages 35–37, New York, NY, USA, October 1997. ACM Press.

[CYWM04] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. Block-based web search. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 456–463, New York, NY, USA, July 2004. ACM Press.

[DALP04] Antoine Doucet, Lili Aunimo, Miro Lehtonen, and Renaud Petit. Accurate Retrieval of XML Document Fragments using EXTIRP. In Fuhr et al. [FLM04], pages 73–80.

[DEG+03] Stephen Dill, Nadav Eiron, David Gibson, Daniel Gruhl, R. Guha, Anant Jhingran, Tapas Kanungo, Sridhar Rajagopalan, Andrew Tomkins, John A. Tomlin, and Jason Y. Zien. Semtag and seeker: bootstrapping the semantic web via automated semantic annotation. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 178–186, New York, NY, USA, May 2003. ACM Press.

[DFF+99] Alin Deutsch, Mary Fernandez, Daniela Florescu, Alon Levy, and Dan Suciu. A query language for XML. In *WWW '99: Proceeding of the eighth international conference on World Wide Web*, pages 1155–1169. ACM Press, May 1999.

[DMD+03] AnHai Doan, Jayant Madhavan, Robin Dhamankar, Pedro Domingos, and Alon Halevy. Learning to match ontologies on the semantic web. *The VLDB Journal*, 12(4):303–319, 2003.

[Dou05] Antoine Doucet. *Advanced Document Description, a Sequential Approach*. PhD thesis, University of Helsinki, November 2005.

[EOY05] Takeharu Eda, Makoto Onizuka, and Masashi Yamamuro. Processing XPath queries with XML summaries. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 223–224, New York, NY, USA, October 2005. ACM Press.

[FG01] Norbert Fuhr and Kai Großjohann. XIRQL: a query language for information retrieval in XML documents. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 172–180. ACM Press, September 2001.

[FGG02] Norbert Fuhr, Norbert Gövert, and Kai Großjohann. HyREX: hyper-media retrieval engine for XML. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, page 449. ACM Press, August 2002.

[FGKL02] Norbert Fuhr, Norbert Gövert, Gabriella Kazai, and Mounia Lalmas, editors. *INEX: Evaluation Initiative for XML retrieval — INEX 2002 Workshop Proceedings*, DELOS Workshop, Schloss Dagstuhl, Germany, December 2002.

[FL04] Norbert Fuhr and Mounia Lalmas. Report on the INEX 2003 Workshop, Schloss Dagstuhl, 15-17 December 2003. *SIGIR FORUM*, 38(1):42–47, June 2004.

[FLM04] Norbert Fuhr, Mounia Lalmas, and Saadia Malik, editors. *INitiative for the Evaluation of XML Retrieval (INEX). Proceedings of the Second INEX Workshop*, Schloss Dagstuhl, Germany, March 2004.

[FLMK06] Norbert Fuhr, Mounia Lalmas, Saadia Malik, and Gabriella Kazai, editors. *Advances in XML Information Retrieval and Evaluation: Fourth International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2005), Dagstuhl, Germany, 28–30 November 2005*, volume 3977 of *Lecture Notes in Computer Science*. Springer, 2006.

[FLMS05] Norbert Fuhr, Mounia Lalmas, Saadia Malik, and Zoltán Szlávik, editors. *Advances in XML Information Retrieval, Third International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2004), Dagstuhl Castle, Germany, 6–8 December 2004, Revised Selected Papers*, volume 3493 of *Lecture Notes in Computer Science*. Springer, 2005.

[GKFL03] Norbert Gövert, Gabriella Kazai, Norbert Fuhr, and Mounia Lalmas. Evaluating the effectiveness of content-oriented XML retrieval. Technical report, University of Dortmund, Computer Science 6, 2003.

[GM05] Robert J. Glushko and Tim McGrath. *Document Engineering*. MIT Press, August 2005.

[Gol03] Charles F. Goldfarb. *The XML Handbook*. Definitive XML Series. Prentice Hall PTR, 5th edition, December 2003.

[GS02] Torsten Grabs and Hans-Jörg Schek. Generating Vector Spaces On-the-fly for Flexible XML Retrieval. In Baeza-Yates et al. [BYFM02a], pages 4–13.

[GW97] Roy Goldman and Jennifer Widom. Dataguides: Enabling query formulation and optimization in

semistructured databases. In *VLDB '97: Proceedings of the 23rd International Conference on Very Large Data Bases*, pages 436–445, San Francisco, CA, USA, August 1997. Morgan Kaufmann Publishers Inc.

[HB03] Jianying Hu and Amit Bagga. Identifying story and preview images in news web pages. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003)*, Edinburgh, Scotland, August 2003. IEEE Computer Society.

[Hei00] Barbara Heikkinen. *Generalization of Document Structures and Document Assembly.* PhD Thesis, Series of Publications A, Report A-2000-2, Department of Computer Science, University of Helsinki, Finland, April 2000.

[HKW+04] Kenji Hatano, Hiroko Kinutani, Masahiro Watanabe, Yasuhiro Mori, Masatoshi Yoshikawa, and Shunsuke Uemura. Keyword-based XML Fragment Retrieval: Experimental Evaluation based on INEX 2003 Relevance Assessments. In Fuhr et al. [FLM04], pages 81–88.

[HLR04] Andreas Henrich, Volker Lüdecke, and Günter Robbert. Applying the IRStream retrieval engine to INEX 2003. In Fuhr et al. [FLM04], pages 118–125.

[HLX03] Ka Kit Hoi, Dik Lun Lee, and Jianliang Xu. Document visualization on small displays. In *Proceedings of the 4th International Conference on Mobile Data Management (MDM 2003)*, pages 262–278, Berlin, Germany, January 2003. Springer-Verlag.

[HY04] Hao He and Jun Yang. Multiresolution Indexing of XML for Frequent Queries. In *ICDE '04: Proceedings of the 20th International Conference on Data Engineering*, pages 683–694, Washington, DC, USA, March 2004. IEEE Computer Society.

[ISO00] ISO/IEC 13249-2:2000. *Information technology — Database languages — SQL Multimedia and Application Packages — Part 2: Full-Text*, International Organization for Standardization, 2000.

[JH01] Euna Jeong and Chun-Nan Hsu. Induction of integrated view for XML data with heterogeneous DTDs. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 151–158, New York, NY, USA, November 2001. ACM Press.

[Jon72] Karen Spärck Jones. A statistical interpretation of term specificity and its application to retrieval. *Journal of Documentation*, 28(1):11–20, March 1972.

[KBNK02] Raghav Kaushik, Philip Bohannon, Jeffrey F Naughton, and Henry F Korth. Covering indexes for branching path queries. In *SIGMOD '02: Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 133–144, New York, NY, USA, June 2002. ACM Press.

[KdRS04] Jaap Kamps, Maarten de Rijke, and Börkur Sigurbjörnsson. Length normalization in XML retrieval. In *SIGIR '04: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 80–87. ACM Press, July 2004.

[KL05] Gabriella Kazai and Mounia Lalmas. Notes on What to Measure in INEX. In *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology*, pages 22–38. Department of Computer Science, University of Otago, Dunedin, New Zealand, 2005.

[KLdV04] Gabriella Kazai, Mounia Lalmas, and Arjen P. de Vries. The overlap problem in content-oriented XML retrieval evaluation. In *SIGIR '04: Proceedings of the 27th annual international conference on Research and development in information retrieval*,

pages 72–79, New York, NY, USA, July 2004. ACM Press.

[Kle99] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[KLM03] Gabriella Kazai, Mounia Lalmas, and Saadia Malik. *INEX'03 Guidelines for Topic Development*, May 2003.

[KSBG02] Raghav Kaushik, Pradeep Shenoy, Philip Bohannon, and Ehud Gudes. Exploiting local similarity for indexing paths in graph-structured data. In *ICDE'02: Proceedings of the 18th International Conference on Data Engineering*, pages 129–140. IEEE Computer Society, February 2002.

[KT00] Mei Kobayashi and Koichi Takeda. Information retrieval on the web. *ACM Computing Surveys (CSUR)*, 32(2):144–173, 2000.

[Lan70] Michael Lane, editor. *Structuralism: a reader*. Jonathan Cape, London, UK, 1970.

[Lar04] Ray R. Larson. Cheshire II at INEX'03: Component and algorithm fusion for XML retrieval. In Fuhr et al. [FLM04], pages 38–45.

[Leh01] Miro Lehtonen. Semi-automatic document assembly with structured source data. Master's thesis, University of Helsinki, 2001.

[Leh05] Miro Lehtonen. EXTIRP 2004: Towards heterogeneity. In Fuhr et al. [FLMS05], pages 372–381.

[LM01] Quanzhong Li and Bongki Moon. Indexing and querying XML data for regular path expressions. In *VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases*, pages 361–370, San Francisco, CA, USA, September 2001. Morgan Kaufmann Publishers Inc.

[LMdV+04] Johan List, Vojkan Mihajlovic, Arjen P. de Vries, Georgina Ramírez, and Djoerd Hiemstra. The TIJAH XML-IR system at INEX 2003. In Fuhr et al. [FLM04], pages 102–109.

[LPHL02] Miro Lehtonen, Renaud Petit, Oskari Heinonen, and Greger Lindén. A dynamic user interface for document assembly. In *Proceedings of the ACM Symposium on Document Engineering*, pages 134–141. ACM Press, November 2002.

[LS81] Geoffrey Leech and Mick Short. *Style in Fiction: A Linguistic Introduction to English Fictional Prose*. English Language Series; 13. Longman, London, UK, 1981.

[Lub04] *HDP '04: Proceedings of the 1st ACM workshop on Hardcopy document processing*, New York, NY, USA, November 2004. ACM Press. General Chair-Kirk Lubbes.

[LW01] Minghorng Lai and Martin D. F. Wong. Slicing tree is a complete floorplan representation. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE 2001)*, pages 228–232. ACM Press, March 2001.

[LZC04] Shaorong Liu, Qinghua Zou, and Wesley W. Chu. Configurable indexing and ranking for XML information retrieval. In *SIGIR '04: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 88–95. ACM Press, July 2004.

[MBHA05] Vojkan Mihajlović;, Henk Ernst Blok, Djoerd Hiemstra, and Peter M. G. Apers. Score region algebra: building a transparent XML-IR database. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 12–19, New York, NY, USA, October 2005. ACM Press.

[MBK91] Yoëlle S. Maarek, Daniel M. Berry, and Gail E. Kaiser. An information retrieval approach for automatically constructing software libraries. *IEEE Trans. Softw. Eng.*, 17(8):800–813, 1991.

[MFRW00] Deborah L. McGuinness, Richard Fikes, James Rice, and Steve Wilder. The Chimaera Ontology Environment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 1123–1124. AAAI Press / The MIT Press, 2000.

[MJCW04] Xiaofeng Meng, Yu Jiang, Yan Chen, and Haixun Wang. Xseq: an indexing infrastructure for tree pattern queries. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 941–942, New York, NY, USA, June 2004. ACM Press.

[MJKZ98] Sung Hyon Myaeng, Dong-Hyun Jang, Mun-Seok Kim, and Zong-Cheol Zhoo. A flexible model for retrieval of SGML documents. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 138–145. ACM Press, August 1998.

[MM04] Yosi Mass and Matan Mandelbrod. Retrieving the most relevant XML Components. In Fuhr et al. [FLM04], pages 53–58.

[MS99] Tova Milo and Dan Suciu. Index structures for path expressions. In *ICDT '99: Proceeding of the 7th International Conference on Database Theory*, pages 277–295, London, UK, January 1999. Springer-Verlag.

[MYTR03] Saikat Mukherjee, Guizhen Yang, Wenfang Tan, and I.V. Ramakrishnan. Automatic discovery of semantic structures in HTML documents. In *Proceedings*

of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003), Edinburgh, Scotland, August 2003. IEEE Computer Society.

[NT80] Makoto Nagao and Jun-Ichi Tsujii. Some topics of language processing for machine translation. *International Forum on Information and Documentation*, 5(2):32–37, 1980.

[NTHK80] Makoto Nagao, Jun-Ichi Tsujii, Hideki Hirakawa, and Masako Kume. A Machine Translation system from Japanese into English - Another Perspective of MT Systems. In *COLING 1980 Volume 1: The 8th International Conference on Computational Linguistics*, pages 414–423, Tokyo, Japan, September 1980.

[NTYK82] Makoto Nagao, Jun-Ichi Tsujii, Koji Yada, and Toshihiro Kakimoto. An English Japanese machine translation system of the titles of scientific and engineering papers. In *Coling 1982: Proceedings of the Ninth International Conference on Computational Linguistics*, pages 245–252, Prague, Czechoslovakia, July 1982.

[PD06] Benjamin Piwowarski and Georges Dupret. Evaluation in (XML) Information Retrieval: Expected Precision-Recall with User Modelling (EPRUM). In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 260–267, New York, NY, USA, August 2006. ACM Press.

[PG03] Benjamin Piwowarski and Patrick Gallinari. Expected ratio of relevant units: A measure for structured information retrieval. In Norbert Fuhr, Mounia Lalmas, and Saadia Malik, editors, *INitiative for the Evaluation of XML Retrieval (INEX). Proceedings of the Second INEX Workshop*, Dagstuhl, Germany, December 2003.

[PK05] K. Hima Prasad and P. Sreenivasa Kumar. Efficient indexing and querying of XML data using modified Prüfer sequences. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 397–404, New York, NY, USA, October 2005. ACM Press.

[PL04a] Benjamin Piwowarski and Mounia Lalmas. Interface pour l'évaluation de systèmes de recherche sur des documents XML. In *Proceedings of Premiere COnference en Recherche d'Information et Applications (CORIA'04)*, pages 109–120. Hermès, March 2004.

[PL04b] Benjamin Piwowarski and Mounia Lalmas. Providing consistent and exhaustive relevance assessments for XML retrieval evaluation. In *CIKM '04: Proceedings of the Thirteenth ACM conference on Information and knowledge management*, pages 361–370, New York, NY, USA, November 2004. ACM Press.

[Por80] Martin F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, July 1980.

[Pri01] Michael Priestley. DITA XML: a reuse by reference architecture for technical documentation. In *SIGDOC '01: Proceedings of the 19th annual international conference on Computer documentation*, pages 152–156, New York, NY, USA, October 2001. ACM Press.

[PVG04] Benjamin Piwowarski, Huyen-Trang Vu, and Patrick Gallinari. Bayesian networks and INEX'03. In Fuhr et al. [FLM04], pages 33–37.

[Rai05] Lee Rainie. Report: The state of blogging. Technical report, PEW Internet & American Life Project, January 2005.

[RILD04] Lakshmish Ramaswamy, Arun Iyengar, Ling Liu, and Fred Douglis. Automatic detection of fragments in dynamically generated web pages. In *WWW '04: Proceedings of the 13th international conference on*

World Wide Web, pages 443–454, New York, NY, USA, May 2004. ACM Press.

[RM04] Praveen Rao and Bongki Moon. PRIX: Indexing And Querying XML Using Prüfer Sequences. In *ICDE '04: Proceedings of the 20th International Conference on Data Engineering*, pages 288–300, Washington, DC, USA, March 2004. IEEE Computer Society.

[SA05] Arnaud Sahuguet and Bogdan Alexe. Sub-Document Queries Over XML with XSQuirrel. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 268–277, New York, NY, USA, May 2005. ACM Press.

[Sal64] Gerard Salton. A document retrieval system for man-machine interaction. In *Proceedings of the 1964 19th ACM national conference*, pages L2.3–1–L2.3–20, New York, NY, USA, 1964. ACM Press.

[SHBM04] Karen Sauvagnat, Gilles Hubert, Mohand Boughanem, and Josiane Mothe. IRIT at INEX 2003. In Fuhr et al. [FLM04], pages 142–148.

[SKdR04] Börkur Sigurbjörnsson, Jaap Kamps, and Maarten de Rijke. An element-based approach to XML retrieval. In Fuhr et al. [FLM04], pages 19–26.

[SL65] Gerard Salton and Michael E. Lesk. The SMART automatic document retrieval systems — an illustration. *Commun. ACM*, 8(6):391–398, 1965.

[SL90] Amit P. Sheth and James A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, September 1990.

[SLWM04] Ruihua Song, Haifeng Liu, Ji-Rong Wen, and Wei-Ying Ma. Learning important models for web page blocks based on layout and content analysis. *SIGKDD Explor. Newsl.*, 6(2):14–23, 2004.

[SR05] Zoltán Szlávik and Thomas Rölleke. Building and experimenting with a heterogeneous collection. In Fuhr et al. [FLMS05], pages 349–357.

[STW04] Ralf Schenkel, Anja Theobald, and Gerhard Weikum. XXL @ INEX 2003. In Fuhr et al. [FLM04], pages 59–66.

[SWY75] Gerard Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.

[TDL+94] Suzanne Liebowitz Taylor, Deborah A. Dahl, Mark Lipshutz, Carl Weir, Lewis M. Norton, Roslyn Nilson, and Marcia Linebarger. Integrated text and image understanding for document understanding. In *HLT '94: Proceedings of the workshop on Human Language Technology*, pages 421–426, Morristown, NJ, USA, March 1994. Association for Computational Linguistics.

[TG06] Andrew Trotman and Shlomo Geva. Passage Retrieval and Other XML-Retrieval Tasks. In *Proceedings of the SIGIR 2006 Workshop on XML Element Retrieval Methodology*, pages 43–50, Dunedin, New Zealand, 2006. University of Otago.

[TLF05] Andrew Trotman, Mounia Lalmas, and Norbert Fuhr, editors. *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology*. Department of Computer Science, University of Otago, July 2005.

[TO04] Andrew Trotman and Richard A. O'Keefe. Identifying and Ranking Relevant Document Elements. In Fuhr et al. [FLM04], pages 149–154.

[TSW05] Martin Theobald, Ralf Schenkel, and Gerhard Weikum. An efficient and versatile query engine for TopX search. In *VLDB '05: Proceedings of the 31st international conference on Very Large Data Bases*, pages 625–636. VLDB Endowment, August 2005.

[Voo94] Ellen M. Voorhees. Query expansion using lexical-semantic relations. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 61–69, New York, NY, USA, July 1994. Springer-Verlag New York, Inc.

[vZWD05] Roelof van Zwol, Frans Wiering, and Virginia Dignum. The Utrecht Blend: Basic Ingredients for an XML Retrieval System. In Fuhr et al. [FLMS05], pages 140–152.

[W3C98] W3C. *Extensible Markup Language (XML) 1.0, W3C Recommendation 10-February-1998*, February 1998. Available at `http://www.w3.org/TR/1998/REC-xml-19980210`.

[W3C01] W3C. *XML Fragment Interchange, W3C Candidate Recommendation 12 February 2001*, 2001. Available at `http://www.w3.org/TR/xml-fragment`.

[W3C02] W3C. *XHTML^{TM}1.0 The Extensible HyperText Markup Language, W3C Recommendation*, 2nd edition, 1 August 2002. Available at `http://www.w3.org/TR/xhtml1/`.

[W3C04] W3C. *XML Information Set, W3C Recommendation*, 2nd edition, 4 February 2004. Latest version available at `http://www.w3.org/TR/xml-infoset/`.

[W3C05a] W3C. *XQuery 1.0: An XML Query Language, W3C Candidate Recommendation*, 3 November 2005. Available at `http://www.w3.org/TR/xquery/`.

[W3C05b] W3C. *XQuery 1.0 and XPath 2.0 Full-Text, W3C Working Draft*, 3 November 2005. Available at `http://www.w3.org/TR/xquery-full-text/`.

[WBD+00] Kevin Williams, Michael Brundage, Patrick Dengler, Jeff Gabriel, Andy Hoskinson, Michael Kay, Thomas Maxwell, Marcelo Ochoa, Johnny Papa, and Mohan

Vanmane. *Professional XML Databases*. Wrox Press Inc., 1st edition, January 2000.

[WD03] Jacky W. W. Wan and Gillian Dobbie. Extracting association rules from XML documents using XQuery. In *WIDM '03: Proceedings of the 5th ACM international workshop on Web information and data management*, pages 94–97. ACM Press, November 2003.

[WDM+02] Louis Weitzman, Sara Elo Dean, Dikran Meliksetian, Kapil Gupta, Nianjun Zhou, and Jessica Wu. Transforming the content management process at ibm.com. In *CHI '02: Case studies of the CHI2002 / AIGA Experience Design FORUM*, pages 1–15, New York, NY, USA, April 2002. ACM Press.

[Wil94] Ross Wilkinson. Effective retrieval of structured documents. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 311–317. ACM Press, August 1994.

[WMSB04] Felix Weigel, Holger Meuss, Klaus U. Schulz, and Fran cois Bry. Content and structure in indexing and ranking XML. In *WebDB '04: Proceedings of the 7th International Workshop on the Web and Databases*, pages 67–72, New York, NY, USA, June 2004. ACM Press.

[WPFY03] Haixun Wang, Sanghyun Park, Wei Fan, and Philip S. Yu. ViST: A dynamic index method for querying XML data by tree structures. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 110–121, New York, NY, USA, June 2003. ACM Press.

[WSM05] Felix Weigel, Klaus U. Schulz, and Holger Meuss. Exploiting native XML indexing techniques for XML retrieval in relational database systems. In *WIDM*

'05: Proceedings of the 7th annual ACM international workshop on Web information and data management, pages 23–30, New York, NY, USA, November 2005. ACM Press.

[WWL+05] Hei Wang, Hongzhi Wang, Hongjun Lu, Haifeng Jiang, Xuemin Lin, and Jianzhong Li. Efficient processing of XML path queries using the disk-based F&B index. In *VLDB '05: Proceedings of the 31st international conference on Very Large Data Bases*, pages 145–156. VLDB Endowment, August 2005.

[XLHF05] Xiangye Xiao, Qiong Luo, Dan Hong, and Hongbo Fu. Slicing*-tree based web page transformation for small displays. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 303–304, New York, NY, USA, October 2005. ACM Press.

[YCWM03] Shipeng Yu, Deng Cai, Ji-Rong Wen, and Wei-Ying Ma. Improving pseudo-relevance feedback in web information retrieval using web page segmentation. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 11–18, New York, NY, USA, May 2003. ACM Press.

[ZA03] Mohammed J. Zaki and Charu C. Aggarwal. XRules: an effective structural classifier for XML data. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 316–325. ACM Press, August 2003.

[ZLC04] Qinghua Zou, Shaorong Liu, and Wesley W. Chu. Ctree: A compact tree for indexing XML data. In *WIDM '04: Proceedings of the 6th annual ACM international workshop on Web information and data management*, pages 39–46, New York, NY, USA, November 2004. ACM Press.

[ZND+01] Chun Zhang, Jeffrey Naughton, David DeWitt, Qiong Luo, and Guy Lohman. On supporting con-

tainment queries in relational database management systems. In *SIGMOD '01: Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, pages 425–436, New York, NY, USA, May 2001. ACM Press.

[ZY04] Ming Zhang and JingTao Yao. XML Algebras for Data Mining. In *Proceedings of SPIE Vol. 5433, Data Mining and Knowledge Discovery: Theory, Tools, and Technology VI*, pages 209–217, April 2004.

CHAPTER A

Big article divided into fragments

```
/article[1]:                          1.16    182497  BIG
/article[1]/fno[1]:                   1.0     5       SMALL
/article[1]/doi[1]:                   1.0     19      SMALL
/article[1]/fm[1]:                    0.98    1824    DATA
/article[1]/fm[1]/hdr[1]:             1.15    129     SMALL
/article[1]/fm[1]/tig[1]:             0.66    60      SMALL
/article[1]/fm[1]/au[1]:              0.57    37      SMALL
/article[1]/fm[1]/au[2]:              0.5     13      SMALL
/article[1]/fm[1]/au[3]:              0.57    23      SMALL
/article[1]/fm[1]/au[4]:              0.57    23      SMALL
/article[1]/fm[1]/au[5]:              0.57    20      SMALL
/article[1]/fm[1]/abs[1]:             1.66    1191    FT-Q
/article[1]/fm[1]/kwd[1]:             1.5     328     FT-Q
/article[1]/bdy[1]:                   1.88    144868  BIG
/article[1]/bdy[1]/sec[1]:            2.75    6602    FT-Q
/article[1]/bdy[1]/sec[2]:            1.68    16255   BIG
/article[1]/bdy[1]/sec[2]/st[1]:      1.0     5       SMALL
/article[1]/bdy[1]/sec[2]/ip1[1]:     1.0     293     FT-Q
/article[1]/bdy[1]/sec[2]/ss1[1]:     1.66    2840    FT-Q
/article[1]/bdy[1]/sec[2]/ss1[2]:     1.66    4325    FT-Q
/article[1]/bdy[1]/sec[2]/ss1[3]:     1.625   4483    FT-Q
/article[1]/bdy[1]/sec[2]/ss1[4]:     1.73    1993    FT-Q
/article[1]/bdy[1]/sec[2]/ss1[5]:     1.57    2316    FT-Q
/article[1]/bdy[1]/sec[3]:            1.95    15039   BIG
/article[1]/bdy[1]/sec[3]/st[1]:      1.66    37      SMALL
/article[1]/bdy[1]/sec[3]/ip1[1]:     1.0     294     FT-Q
/article[1]/bdy[1]/sec[3]/p[1]:       1.5     291     FT-Q
/article[1]/bdy[1]/sec[3]/tf[1]:      1.0     36      SMALL
/article[1]/bdy[1]/sec[3]/en[1]:      1.0     3       SMALL
/article[1]/bdy[1]/sec[3]/ip1[2]:     1.90    397     FT-Q
/article[1]/bdy[1]/sec[3]/fig[1]:     1.0     86      SMALL
/article[1]/bdy[1]/sec[3]/p[2]:       1.0     158     FT-Q
/article[1]/bdy[1]/sec[3]/p[3]:       1.75    660     FT-Q
/article[1]/bdy[1]/sec[3]/tf[2]:      1.0     97      SMALL
/article[1]/bdy[1]/sec[3]/en[2]:      1.0     3       SMALL
/article[1]/bdy[1]/sec[3]/ip1[3]:     1.83    127     SMALL
/article[1]/bdy[1]/sec[3]/tf[3]:      1.0     80      SMALL
```

```
/article[1]/bdy[1]/sec[3]/en[3]:            1.0     3       SMALL
/article[1]/bdy[1]/sec[3]/p[4]:             1.875   1335    FT-Q
/article[1]/bdy[1]/sec[3]/ss1[1]:           1.94    5071    FT-Q
/article[1]/bdy[1]/sec[3]/ss1[2]:           2.375   1768    FT-Q
/article[1]/bdy[1]/sec[3]/ss1[3]:           1.92    2312    FT-Q
/article[1]/bdy[1]/sec[3]/ss1[4]:           1.8     2281    FT-Q
/article[1]/bdy[1]/sec[4]:                  1.87    24057   BIG
/article[1]/bdy[1]/sec[4]/st[1]:            1.6     30      SMALL
/article[1]/bdy[1]/sec[4]/ip1[1]:           1.0     257     FT-Q
/article[1]/bdy[1]/sec[4]/ss1[1]:           1.76    3756    FT-Q
/article[1]/bdy[1]/sec[4]/ss1[2]:           1.93    5728    FT-Q
/article[1]/bdy[1]/sec[4]/ss1[3]:           1.86    2395    FT-Q
/article[1]/bdy[1]/sec[4]/ss1[4]:           1.83    1131    FT-Q
/article[1]/bdy[1]/sec[4]/ss1[5]:           1.89    3987    FT-Q
/article[1]/bdy[1]/sec[4]/ss1[6]:           1.88    2138    FT-Q
/article[1]/bdy[1]/sec[4]/ss1[7]:           1.75    4635    FT-Q
/article[1]/bdy[1]/sec[5]:                  1.93    23915   BIG
/article[1]/bdy[1]/sec[5]/st[1]:            1.5     27      SMALL
/article[1]/bdy[1]/sec[5]/ip1[1]:           1.5     319     FT-Q
/article[1]/bdy[1]/sec[5]/fig[1]:           0.75    60      SMALL
/article[1]/bdy[1]/sec[5]/ss1[1]:           1.95    2260    FT-Q
/article[1]/bdy[1]/sec[5]/ss1[2]:           1.96    5434    FT-Q
/article[1]/bdy[1]/sec[5]/ss1[3]:           1.85    1834    FT-Q
/article[1]/bdy[1]/sec[5]/ss1[4]:           1.89    2568    FT-Q
/article[1]/bdy[1]/sec[5]/ss1[5]:           1.89    4797    FT-Q
/article[1]/bdy[1]/sec[5]/ss1[6]:           1.83    1537    FT-Q
/article[1]/bdy[1]/sec[5]/ss1[7]:           2.09    2061    FT-Q
/article[1]/bdy[1]/sec[5]/ss1[8]:           1.71    3018    FT-Q
/article[1]/bdy[1]/sec[6]:                  1.88    23094   BIG
/article[1]/bdy[1]/sec[6]/st[1]:            1.0     11      SMALL
/article[1]/bdy[1]/sec[6]/ip1[1]:           1.5     577     FT-Q
/article[1]/bdy[1]/sec[6]/ss1[1]:           1.91    2993    FT-Q
/article[1]/bdy[1]/sec[6]/ss1[2]:           1.86    6911    FT-Q
/article[1]/bdy[1]/sec[6]/ss1[3]:           1.91    2948    FT-Q
/article[1]/bdy[1]/sec[6]/ss1[4]:           1.84    6187    FT-Q
/article[1]/bdy[1]/sec[6]/ss1[5]:           1.71    3467    FT-Q
/article[1]/bdy[1]/sec[7]:                  1.95    17913   BIG
/article[1]/bdy[1]/sec[7]/st[1]:            1.33    13      SMALL
/article[1]/bdy[1]/sec[7]/ss1[1]:           1.97    7777    FT-Q
/article[1]/bdy[1]/sec[7]/ss1[2]:           2.0     4559    FT-Q
/article[1]/bdy[1]/sec[7]/ss1[3]:           1.97    4561    FT-Q
/article[1]/bdy[1]/sec[7]/ss1[4]:           1.25    1003    FT-Q
/article[1]/bdy[1]/sec[8]:                  1.44    17993   BIG
/article[1]/bdy[1]/sec[8]/st[1]:            1.33    17      SMALL
/article[1]/bdy[1]/sec[8]/ip1[1]:           1.0     77      SMALL
/article[1]/bdy[1]/sec[8]/list[1]:          1.43    17899   BIG
/article[1]/bdy[1]/sec[8]/list[1]/item[1]:  1.28    1650    FT-Q
/article[1]/bdy[1]/sec[8]/list[1]/item[2]:  1.44    2007    FT-Q
/article[1]/bdy[1]/sec[8]/list[1]/item[3]:  1.44    3969    FT-Q
/article[1]/bdy[1]/sec[8]/list[1]/item[4]:  1.375   2606    FT-Q
/article[1]/bdy[1]/sec[8]/list[1]/item[5]:  1.375   2206    FT-Q
/article[1]/bdy[1]/sec[8]/list[1]/item[6]:  1.9     1818    FT-Q
/article[1]/bdy[1]/sec[8]/list[1]/item[7]:  1.375   2524    FT-Q
/article[1]/bdy[1]/sec[8]/list[1]/item[8]:  1.28    1119    FT-Q
/article[1]/bm[1]:                          0.88    35781   BIG
```

```
/article[1]/bm[1]/ack[1]:                          1.25    350     FT-Q
/article[1]/bm[1]/footnote[1]:                     1.0     819     FT-Q
/article[1]/bm[1]/footnote[2]:                     1.5     55      SMALL
/article[1]/bm[1]/bib[1]:                          0.87    30083   BIG
/article[1]/bm[1]/bib[1]/bibl[1]:                  0.87    30083   BIG
/article[1]/bm[1]/bib[1]/bibl[1]/h[1]:             1.0     10      SMALL
/article[1]/bm[1]/bib[1]/bibl[1]/h[1]/scp[1]:      1.0     9       SMALL
/article[1]/bm[1]/bib[1]/bibl[1]/bb[1]:            0.77    175     DATA
/article[1]/bm[1]/bib[1]/bibl[1]/bb[1]/au[1]:      0.66    9       SMALL
/article[1]/bm[1]/bib[1]/bibl[1]/bb[1]/au[2]:      0.66    7       SMALL
/article[1]/bm[1]/bib[1]/bibl[1]/bb[1]/obi[1]:     1.0     3       SMALL
/article[1]/bm[1]/bib[1]/bibl[1]/bb[1]/au[3]:      0.66    10      SMALL
/article[1]/bm[1]/bib[1]/bibl[1]/bb[1]/atl[1]:     3.0     88      SMALL
/article[1]/bm[1]/bib[1]/bibl[1]/bb[1]/ti[1]:      1.0     34      SMALL
/article[1]/bm[1]/bib[1]/bibl[1]/bb[1]/obi[2]:     0.5     7       SMALL
/article[1]/bm[1]/bib[1]/bibl[1]/bb[1]/pp[1]:      1.0     12      SMALL
/article[1]/bm[1]/bib[1]/bibl[1]/bb[1]/pdt[1]:     0.5     5       SMALL
/article[1]/bm[1]/bib[1]/bibl[1]/bb[2]:            0.84    143     SMALL
/article[1]/bm[1]/bib[1]/bibl[1]/bb[3]:            0.82    172     DATA
...
/article[1]/bm[1]/bib[1]/bibl[1]/bb[44]:           1.0     180   FT-Q
...
/article[1]/bm[1]/bib[1]/bibl[1]/bb[199]:          0.83    115     SMALL
/article[1]/bm[1]/bib[1]/bibl[1]/bb[200]:          0.78    120     SMALL
/article[1]/bm[1]/bib[1]/bibl[1]/bb[201]:          0.82    157     DATA
/article[1]/bm[1]/bib[1]/bibl[1]/bb[201]/au[1]:    0.66    5       SMALL
/article[1]/bm[1]/bib[1]/bibl[1]/bb[201]/obi[1]:   1.0     3       SMALL
/article[1]/bm[1]/bib[1]/bibl[1]/bb[201]/au[2]:    0.66    12      SMALL
/article[1]/bm[1]/bib[1]/bibl[1]/bb[201]/atl[1]:   3.0     48      SMALL
/article[1]/bm[1]/bib[1]/bibl[1]/bb[201]/ti[1]:    1.0     54      SMALL
/article[1]/bm[1]/bib[1]/bibl[1]/bb[201]/obi[2]:   0.66    14      SMALL
/article[1]/bm[1]/bib[1]/bibl[1]/bb[201]/pp[1]:    1.0     12      SMALL
/article[1]/bm[1]/bib[1]/bibl[1]/bb[201]/pdt[1]:   0.66    9       SMALL
/article[1]/bm[1]/vt[1]:                           1.5     940     FT-Q
/article[1]/bm[1]/vt[2]:                           1.2     623     FT-Q
/article[1]/bm[1]/vt[3]:                           1.33    812     FT-Q
/article[1]/bm[1]/vt[4]:                           1.2     742     FT-Q
/article[1]/bm[1]/vt[5]:                           1.44    1357    FT-Q
```

TIETOJENKÄSITTELYTIETEEN LAITOS
PL 68 (Gustaf Hällströmin katu 2 b)
00014 Helsingin yliopisto

DEPARTMENT OF COMPUTER SCIENCE
P.O. Box 68 (Gustaf Hällströmin katu 2 b)
FIN-00014 University of Helsinki, FINLAND

JULKAISUSARJA **A**

SERIES OF PUBLICATIONS **A**

Reports may be ordered from: Kumpula Science Library, P.O. Box 64, FIN-00014 University of Helsinki, FINLAND.

A-1992-1 J. Kivinen: Problems in computational learning theory. 27 + 64 pp. (Ph.D. thesis).

A-1992-2 K. Pohjonen & J. Tarhio (toim./eds.):

A-1992-3 Th. Eiter, P. Kilpeläinen & H. Mannila: Recognizing renamable generalized propositional Horn formulas is NP-complete. 11 pp.

A-1992-4 A. Valmari: Alleviating state explosion during verification of behavioural equivalence. 57 pp.

A-1992-5 P. Floréen: Computational complexity problems in neural associative memories. 128 + 8 pp. (Ph.D. thesis).

A-1992-6 P. Kilpeläinen: Tree matching problems with applications to structured text databases. 110 pp. (Ph.D. thesis).

A-1993-1 E. Ukkonen: On-line construction of suffix-trees. 15 pp.

A-1993-2 Alois P. Heinz: Efficient implementation of a neural net α-β-evaluator. 13 pp.

A-1994-1 J. Eloranta: Minimal transition systems with respect to divergence preserving behavioural equivalences. 162 pp. (Ph.D. thesis).

A-1994-2 K. Pohjonen (toim./ed.): Tietojenkäsittelyopin laitoksen julkaisut 1992–93 – Publications from the Department of Computer Science 1992–93. 58 s./pp.

A-1994-3 T. Kujala & M. Tienari (eds.): Computer Science at the University of Helsinki 1993. 95 pp.

A-1994-4 P. Floréen & P. Orponen: Complexity issues in discrete Hopfield networks. 54 pp.

A-1995-1 P. Myllymäki: Mapping Bayesian networks to stochastic neural networks: a foundation for hybrid Bayesian-neural systems. 93 pp. (Ph.D. thesis).

A-1996-1 R. Kaivola: Equivalences, preorders and compositional verification for linear time temporal logic and concurrent systems. 185 pp. (Ph.D. thesis).

A-1996-2 T. Elomaa: Tools and techniques for decision tree learning. 140 pp. (Ph.D. thesis).

A-1996-3 J. Tarhio & M. Tienari (eds.): Computer Science at the University of Helsinki 1996. 89 pp.

A-1996-4 H. Ahonen: Generating grammars for structured documents using grammatical inference methods. 107 pp. (Ph.D. thesis).

A-1996-5 H. Toivonen: Discovery of frequent patterns in large data collections. 116 pp. (Ph.D. thesis).

A-1997-1 H. Tirri: Plausible prediction by Bayesian inference. 158 pp. (Ph.D. thesis).

A-1997-2 G. Lindén: Structured document transformations. 122 pp. (Ph.D. thesis).

A-1997-3 M. Nykänen: Querying string databases with modal logic. 150 pp. (Ph.D. thesis).

A-1997-4 E. Sutinen, J. Tarhio, S.-P. Lahtinen, A.-P. Tuovinen, E. Rautama & V. Meisalo: Eliot – an algorithm animation environment. 49 pp.

A-1998-1 G. Lindén & M. Tienari (eds.): Computer Science at the University of Helsinki 1998. 112 pp.

A-1998-2 L. Kutvonen: Trading services in open distributed environments. 231 + 6 pp. (Ph.D. thesis).

A-1998-3 E. Sutinen: Approximate pattern matching with the q-gram family. 116 pp. (Ph.D. thesis).

A-1999-1 M. Klemettinen: A knowledge discovery methodology for telecommunication network alarm databases. 137 pp. (Ph.D. thesis).

A-1999-2 J. Puustjärvi: Transactional workflows. 104 pp. (Ph.D. thesis).

A-1999-3 G. Lindén & E. Ukkonen (eds.): Department of Computer Science: annual report 1998. 55 pp.

A-1999-4 J. Kärkkäinen: Repetition-based text indexes. 106 pp. (Ph.D. thesis).

A-2000-1 P. Moen: Attribute, event sequence, and event type similarity notions for data mining. 190+9 pp. (Ph.D. thesis).

A-2000-2 B. Heikkinen: Generalization of document structures and document assembly. 179 pp. (Ph.D. thesis).

A-2000-3 P. Kähkipuro: Performance modeling framework for CORBA based distributed systems. 151+15 pp. (Ph.D. thesis).

A-2000-4 K. Lemström: String matching techniques for music retrieval. 56+56 pp. (Ph.D.Thesis).

A-2000-5 T. Karvi: Partially defined Lotos specifications and their refinement relations. 157 pp. (Ph.D.Thesis).

A-2001-1 J. Rousu: Efficient range partitioning in classification learning. 68+74 pp. (Ph.D. thesis)

A-2001-2 M. Salmenkivi: Computational methods for intensity models. 145 pp. (Ph.D. thesis)

A-2001-3 K. Fredriksson: Rotation invariant template matching. 138 pp. (Ph.D. thesis)

A-2002-1 A.-P. Tuovinen: Object-oriented engineering of visual languages. 185 pp. (Ph.D. thesis)

A-2002-2 V. Ollikainen: Simulation techniques for disease gene localization in isolated populations. 149+5 pp. (Ph.D. thesis)

A-2002-3 J. Vilo: Discovery from biosequences. 149 pp. (Ph.D. thesis)

A-2003-1 J. Lindström: Optimistic concurrency control methods for real-time database systems. 111 pp. (Ph.D. thesis)

A-2003-2 H. Helin: Supporting nomadic agent-based applications in the FIPA agent architecture. 200+17 pp. (Ph.D. thesis)

A-2003-3 S. Campadello: Middleware infrastructure for distributed mobile applications. 164 pp. (Ph.D. thesis)

A-2003-4 J. Taina: Design and analysis of a distributed database architecture for IN/GSM data. 130 pp. (Ph.D. thesis)

A-2003-5 J. Kurhila: Considering individual differences in computer-supported special and elementary education. 135 pp. (Ph.D. thesis)

A-2003-6 V. Mäkinen: Parameterized approximate string matching and local-similarity-based point-pattern matching. 144 pp. (Ph.D. thesis)

A-2003-7 M. Luukkainen: A process algebraic reduction strategy for automata theoretic verification of untimed and timed concurrent systems. 141 pp. (Ph.D. thesis)

A-2003-8 J. Manner: Provision of quality of service in IP-based mobile access networks. 191 pp. (Ph.D. thesis)

A-2004-1 M. Koivisto: Sum-product algorithms for the analysis of genetic risks. 155 pp. (Ph.D. thesis)

A-2004-2 A. Gurtov: Efficient data transport in wireless overlay networks. 141 pp. (Ph.D. thesis)

A-2004-3 K. Vasko: Computational methods and models for paleoecology. 176 pp. (Ph.D. thesis)

A-2004-4 P. Sevon: Algorithms for Association-Based Gene Mapping. 101 pp. (Ph.D. thesis)

A-2004-5 J. Viljamaa: Applying Formal Concept Analysis to Extract Framework Reuse Interface Specifications from Source Code. 206 pp. (Ph.D. thesis)

A-2004-6 J. Ravantti: Computational Methods for Reconstructing Macromolecular Complexes from Cryo-Electron Microscopy Images. 100 pp. (Ph.D. thesis)

A-2004-7 M. Kääriäinen: Learning Small Trees and Graphs that Generalize. 45+49 pp. (Ph.D. thesis)

A-2004-8 T. Kivioja: Computational Tools for a Novel Transcriptional Profiling Method. 98 pp. (Ph.D. thesis)

A-2004-9 H. Tamm: On Minimality and Size Reduction of One-Tape and Multitape Finite Automata. 80 pp. (Ph.D. thesis)

A-2005-1 T. Mielikäinen: Summarization Techniques for Pattern Collections in Data Mining. 201 pp. (Ph.D. thesis)

A-2005-2 A. Doucet: Advanced Document Description, a Sequential Approach. 161 pp. (Ph.D. thesis)

A-2006-1 A. Viljamaa: Specifying Reuse Interfaces for Task-Oriented Framework Specialization 179+107 pp. (Ph.D. thesis)

A-2006-2 S. Tarkoma: Efficient Content-based Routing, Mobility-aware Topologies, and Temporal Subspace Matching 192+6 pp. (Ph.D. thesis)