Theoretical Evaluation of XML Retrieval

Tobias Blanke

Department of Computing Science

University of Glasgow

A thesis submitted for the degree of

Doctor of Philosophy

Acknowledgements

First I would like to acknowledge my PhD supervisors, Mounia Lalmas and Keith van Rijsbergen. Mounia has worked with me, ever since I started this thesis based on a Master dissertation I did with her at Queen Mary University London. Without her, this thesis would have never been possible, which is in my opinion the best possible compliment one can make one's first supervisor. Keith has given me valuable input towards the end of my thesis, which increased my focus and helped me overcome those final obstacles that seem to be the hardest. I would also like to thank those people whose thoughts made my thinking about theoretical evaluation possible. Next to Mounia and Keith, I owe most to Theo Huibers for his work on the use of Situation Theory for a theoretical evaluation. Dawei Song and Peter Bruza both gave me excellent feedback after a conference presentation. My examiners have both helped to iron out remaining inconsistencies. I should especially mention Paul Cockshott from Glasgow who as an internal has added valuable insights from a perspective outside information retrieval research. My fellow PhD students from two top quality information retrieval research groups in London and Glasgow were all happy to engage with my rather unusual topic and give me new ideas and feedback. Finally, I would like to thank my partner Claudia who has been an example to me in more than one sense, as the reader of this thesis will find out.

Abstract

This thesis develops a theoretical framework to evaluate XML retrieval. XML retrieval deals with retrieving those document parts that specifically answer a query. It is concerned with using the document structure to improve the retrieval of information from documents by only delivering those parts of a document an information need is about. We define a theoretical evaluation methodology based on the idea of 'aboutness' and apply it to XML retrieval models. Situation Theory is used to express the aboutness proprieties of XML retrieval models. We propose a Situation Theory framework to evaluate XML retrieval, which is based on the basic and most general information retrieval question how a document (or in our case an XML element) can be about a query. This framework allows us to compare and analyze the reasoning behaviour of XML retrieval models experimented within INEX evaluation campaigns. We develop a dedicated methodology for the evaluation of XML retrieval and apply this methodology to five XML retrieval models from INEX. For each model we derive functional and qualitative properties that qualify its formal behaviour. We compare this behaviour with the underlying flat document retrieval model as well as with a model we specially design to determine how much an XML retrieval model includes XML structure in its reasoning behaviour. More INEX specific, this thesis further investigates the use of our theoretical evaluation methodology to describe the INEX evaluation methodology. We exemplify theoretical models of user agents and assessment procedures in INEX and derive reasoning assumptions that are included in the specific XML retrieval experimental evaluation, its scales and the ways assessments are done. We point to potential inconsistencies and make suggestions for alternative views on the experimental evaluation dimensions for XML retrieval. Further INEX specifics are discussed when we theoretically analyse filters, as they are used in INEX to deliver only specific answers to an information need. We introduce our theoretical methodology to analyse filters as special aboutness decisions, before applying it to the XML retrieval filtering models. We finally use the theoretical properties of XML retrieval models and their filters to explain experimental results obtained with some of the XML retrieval models within INEX and draw upon all our previous results to demonstrate how theoretical evaluation insights can be used to explain results from mainstream experimental evaluations. We relate our theoretical evaluation results with the experimental ones for XML retrieval to find out how the adjustment of existing flat document retrieval models compares to the creation of completely

new ones, especially designed to meet the requirements of XML retrieval. For each of the XML retrieval experimental evaluation tasks, we shall determine the reasoning properties that support a good performance and discuss on this basis the experimental performance of the XML retrieval models

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The general information retrieval problem is to find a set of information sources (mainly documents) relevant to an information need (commonly expressed in a query). The aim of XML retrieval is to retrieve relevant document parts at the right level of granularity, i.e. those that specifically answer a query. XML, contrary to HTML, separates the logical structure of documents from the layout. XML retrieval is concerned with using this document structure to improve the retrieval of information from documents by only delivering those parts of a document that are the most relevant ones to an information need.

For XML retrieval, the information sources are documents structured with XML, while the queries can also contain structural hints or be purely content-based. Given that XML has become an accepted standard method to structure information, making it easier for applications and devices of all kinds to use and store information, XML retrieval has also become an increasingly important research topic. This thesis attempts a theoretical evaluation of XML retrieval and develops a general framework to do so.

Theoretical evaluation of information retrieval is concerned with the formal representation of retrieval models, which includes the symbolic representation of the way a retrieval model captures information and the analysis of the matching function between information need and document. A theoretical evaluation is then complementary to an experimental evaluation if it helps to clarify the assumptions of retrieval models and if it can identify the characteristics leading to a particular experimental behaviour.

In this work, we define a theoretical methodology based on the idea of 'aboutness' and apply it to XML retrieval models. Our framework to evaluate XML retrieval is based on the basic and most general information retrieval question: how a document (in our case an XML-structured one) can be *about* a query. This allows us to compare and analyse the behaviour of XML retrieval models.

As we consider aboutness as the cornerstone of our theoretical evaluation, we need a way of expressing aboutness relations. To this end, the thesis draws on existing logic-based approaches in order to derive the properties of aboutness relations and to analyse information retrieval processes. It will become clear in our thesis that an aboutness-based theoretical evaluation can complement mainstream experimental evaluations and provides an opportunity to produce new and redevelop traditional IR models. This is especially important for the more complex IR tasks of the future, one of which is the effective retrieval from XML document repositories.

Our general hypothesis is that particularly in the domain of XML retrieval, an aboutness-based theoretical evaluation presents a powerful methodology to analyse the complex interaction of XML structure and content. This thesis will adjust existing aboutness-based evaluation approaches to reflect the requirements of XML retrieval and develop a new methodology. Based on this new methodology we will be able to theoretically evaluate various XML retrieval models and underlying aboutness assumptions of specifics of XML retrieval such as experimental evaluation strategies. We will argue that our theoretical evaluation leads to a better understanding of a model's retrieval performance in the experimental evaluation.

The thesis is organised in three larger parts. The first one introduces the background, the second develops our methodology, while the third applies the methodology to analyse XML retrieval models and further specifics of XML retrieval such as its experimental evaluation strategies.

Chapter 2 is the first of two background chapters introducing XML retrieval in more depth and our choice for a theoretical evaluation framework. It introduces basic XML concepts relevant to this thesis and finally describes the subdiscipline of information retrieval called XML retrieval. It details some of the history of XML retrieval as well as XML retrieval approaches that are emerging in this field and evaluation methodologies that have been developed to support the specific aims of XML retrieval.

While the analysis of XML retrieval is well developed by now, there has been little, if any, work done on a systematic theoretical evaluation. This thesis undertakes such a theoretical evaluation. For this purpose, the second background chapter 3 introduces the concept of theoretical evaluation, its history, predominant approaches and finally our choice of using an aboutness-based approach to theoretically analyse XML retrieval. We demonstrate why a theoretical evaluation based on aboutness is needed, as it helps reveal some of the underlying assumptions of the XML retrieval work done so far.

In Chapter 3, we also introduce the framework we need to execute our theoretical evaluation based on the logical analysis of reasoning processes involved in XML retrieval models. We build upon an advanced mathematisation of natural language semantics called Situation Theory. We introduce its basic concepts and show how its ontology is particularly well suited for the analysis of information retrieval models. We derive what an aboutness analysis based on Situation Theory means for XML retrieval in particular. We introduce XML structure to aboutness research, as a component of the aboutness decision and develop a framework for this, so that at the end of this chapter we are able to generally define XML retrieval aboutness and redefine existing approaches to Situation Theory aboutness, which allow for the inclusion of structure.

While Chapters 2 and 3 offer the background, Chapter 4 presents first results of our work. Our research into developing an aboutness-based theoretical evaluation of XML retrieval commences with the development of our methodology, where we adjust existing theoretical evaluation methodologies to the requirements of XML retrieval, analyse gaps and finally amend existing approaches with new parts that help describe the reasoning behaviour of XML retrieval models. We develop our theoretical evaluation steps and present with pure type XML retrieval a means to measure the impact of XML structure on the reasoning behaviour of XML retrieval models.

Our evaluation methodology stands in the tradition of theoretical evaluation as it uses a well-defined number of steps to firstly symbolically represent an XML retrieval model and to secondly analyse its functional behaviour using logical reasoning rules. Each of our theoretical evaluations of XML retrieval models goes through the same four steps to define the characteristics of a particular XML retrieval model.

The first step is the translation of the way the model indexes information into a symbolic representation, which we can use in the second step to analyse the aboutness definition with reasoning rules. In Section 4.4, we define reasoning rules to describe the functional behaviour of XML retrieval models. Our discussion of aboutness rules covers basic rules, then combination and containment rules before we finish by discussing how non-aboutness reasoning can enhance IR models. Some of the rules that are key to our theoretical evaluation are presented. These include traditional reasoning rules such as Symmetry and Transitivity as well as variants of combination rules expressing monotonic reasoning, which have proven to be particularly important and conclusive for the analysis of XML retrieval models. Furthermore, containment rules express important characteristics of XML documents. By comparing the kind of rules a particular model incorporates and the way it does so, we are able to provide an analysis of the behaviour of XML retrieval models.

A further investigation of aboutness boundaries for particular retrieval models, which we call reflection, is the third step of the theoretical evaluation. The final step in our theoretical evaluation is the comparison with the pure type XML retrieval model to analyse the impact of structure on retrieval performance. This forth step is an addition to the existing aboutness-based theoretical evaluation frameworks.

In Chapter 5, we apply our theoretical evaluation methodology to five successful XML retrieval models. We present an XML vector space model, two XML language modelling models and two further ones, which have been specifically designed for XML retrieval. For each of these models, we go through all of the theoretical evaluation methodology steps from Chapter 4 and draw conclusions on the representation of information in the model as well as its reasoning behaviour. We are particularly interested in discussing how standard information retrieval models such as vector space retrieval have been changed to meet the requirements of XML retrieval and what kind of assumptions have guided the development of new models for XML retrieval.

Chapter 6 adds another new dimension to a theoretical evaluation based on aboutness. Aided by the fact that a new systematic experimental evaluation framework has been developed for XML retrieval, we formulate a theoretical evaluation of the experimental evaluation for XML retrieval. We derive reasoning assumptions that are included in the specific XML retrieval experimental evaluation, its scales and the ways assessments are done. We point to potential inconsistencies and make suggestions for alternative views on the experimental evaluation dimensions for XML retrieval. We demonstrate so-called agent reasoning models reflecting the various user interests expressed in XML retrieval evaluation methodologies and use this to theoretically evaluate the experimental evaluation.

We continue with our theoretical evaluation work specific to XML retrieval, when in Chapter 7 we analyse XML retrieval filtering used to deliver only answers that are most specific to an information need. We introduce a new theoretical methodology to analyse filters as aboutness decisions, before applying it to the XML retrieval filtering models. Two main types of filters have been used in XML retrieval: a simple brute-force filter that only keeps the highest ranked element of each XML path and a more complex one that takes into account the relations between retrieved XML elements. For the latter, we will look at the Utility Prior filter and the re-ranking approach. The first one takes into account the utility of an existing XML element, while the second one uses the direct relationships of an XML element to re-rank it.

Finally, Chapter 8 draws upon all the results from the previous chapters to demonstrate how theoretical evaluation insights can be used to explain results from mainstream experimental evaluations. It relates our theoretical evaluation results with the experimental ones for XML retrieval to find out how the adjustment of existing flat document retrieval models compares to the creation of completely new ones, especially designed to meet the requirements of XML retrieval. For each of the XML retrieval experimental evaluation tasks, we shall determine the reasoning properties that support a good performance and discuss the experimental performance of the XML retrieval models from Chapter 5. To our knowledge, no existing aboutness approach has actually delivered such an in-depth analysis of experimental results using the insights from the theoretical evaluation.

Our conclusion summarises the results and contributions of the thesis and considers critically advantages and limitations of our approach as well as possibilities for future work.

Chapter 2

XML Retrieval

2.1 XML Retrieval in context

XML retrieval is a recent development in information retrieval (IR) [Lalmas, 2009]. It is frequently referred to as structured document retrieval [Manning et al., 2008], although this characterisation might be misleading as structured retrieval is often associated with database retrieval, for which individual records are retrieved from tables in databases using a dedicated structured query language. Compared to this database task, XML retrieval is not structured document retrieval, as its aim is to retrieve from unstructured information — commonly from texts as in XML retrieval but more and more also from other multimedia content. However, compared to general IR, XML retrieval is structured document retrieval, as it uses structure of documents to improve the retrieval results. The context of XML retrieval is therefore its distinction from traditional IR on the one hand side and from database retrieval on the other [Lalmas and Baeza-Yates, 2010].

IR systems are often distinguied from traditional databases in terms of the objects both contain. The former are considered to look at unstructured information while the latter query structured information, following a relational calculus [Manning et al., 2008]. According to this distinction, searching in databases targets 'records' looked up in database tables, while in IR, searching aims for 'documents' looked up in indexes. Database systems look for data, while IR systems target information. But nowadays, the difference between databases and IR systems is no longer a question of the objects they contain. Databases can contain documents, too. In fact, many modern database management systems define special fields to cover texts (and other multimedia documents) and might offer special indexes to search them. However, most textual information is still considered to be better modelled outside database systems [Manning et al., 2008, p. 195], as they are often not just for consumption by computers but for consumption by humans, too. XML seems to be the most common choice for structuring texts outside database systems. That is why delivering XML retrieval solutions is so important.

In order to distinguish database retrieval from information retrieval, it is useful to start with their different objectives. The general process of IR [van Rijsbergen, 1979] commences with an information need, which a user expresses as queries to an IR system. Such a query is normally entered as informal expressions; for example, as query terms into search fields of web search engines. IR processes are therefore characterised by the complex interaction between an information need and a system's response. This complex interaction differs in two aspects (among others) from the experience in traditional database systems:

- The IR user does not have to express her information need in a formal language.
- It is the assumption that an IR system only represents parts of the information of the documents in its scope. The documents generally contain more information than what the system is able to represent and present.

This means that several documents may be a match to a query as an expression of an information need. IR systems then compute a score on how 'relevant' a document is to this ‘information need’ and return the documents ranked by these scores [van Rijsbergen, 1979].

Such a ranked list of documents is delivered in two fundamental processing steps. In the first so-called indexing step, an IR system aims to find a representation of the information that models the available information content in the considered documents, while also delivering a computationally viable representation. In the second step, in the actual retrieval of documents, the IR system computes a retrieval status value of how well each document according to its index matches the information need in a query, and ranks the objects according to this value. The user is then presented with a choice of those objects in the index that most likely correspond to her information need.

Generally speaking, an IR system is software that helps its users to find the information they need. It fulfills this information need not by delivering the information directly but by pointing the users to the location of possible information sources (mostly documents). The IR system will suggest possible answers. For the IR system, a document has not got to be an answer to a query. It assigns weights as a measure how ‘likely’ a document might be an answer to a query. Relevance measured by assigning weights is therefore key in IR and denotes how well a user’s information need is met. In order to deliver an effective weighting, an IR system will use all the information it can process, which might also include the structure of documents that can be found in their XML representation, or their associated metadata, etc.

An IR system tries to satisfy a user’s information need by interpreting the contents of information objects, which may involve the interpretation of the syntactic and semantic information in the document [Baeza-Yates and Ribeiro-Neto, 1999]. Structured document retrieval aims to use the structure of the information to improve the retrieval of information. For instance, instead of returning a whole book only a particularly relevant chapter will be returned. Or, if a title element is about an information need, it can be assumed that the following document parts are highly relevant to the information need. Structured document retrieval is concerned with the development of models for querying and retrieving from structured information [Manning et al., 2008]. XML retrieval can be seen as a special case of structured document retrieval for texts marked up in XML.

XML retrieval is the attempt to use XML structure to improve the delivery of relevant information from XML document repositories. Structure has always been very important for the development of IR models. The document length, for instance, has proven to influence the overall performance of IR models [Manning et al., 2008]. Length is only a simple example of structure that influences the results of an IR process. Modern applications in structured document retrieval also include Internet applications that provide users with structured information about their content such as blogs or wikis, and finally modern word processing and other office technologies, which capture their texts, etc. in standardised XML outputs [Manning et al., 2008].

It may be possible to call such IR practices based on structure semi-structured retrieval techniques [Lalmas and Baeza-Yates, 2010] rather than structured retrieval, which would distinguish them better from database retrieval. Yet, this is not the agreed terminology. When talking about structured retrieval, most researchers refer to the task of using structure to improve the results of textual retrieval. Structured document retrieval was first introduced in the mid 1990's and has a history of various meanings from hypermedia retrieval, passage retrieval to XML retrieval. We look at this history in Section 3.3, as it provides a good background for the theoretical understanding of XML retrieval, the subject of this thesis.

In this chapter we briefly introduce the concepts and ideas behind XML retrieval. In the next section, we will discuss the document format XML (eXtensible Markup Language), a W3C standard for marking up texts.[1]

2.2 XML and Related Concepts

XML retrieval systems aim to provide effective access to XML document repositories. XML is a simplified version of the earlier standard of SGML, which itself stands for Standard Generalized Markup Language [Manning et al., 2008]. Just like its predecessor SGML, XML is a meta-language that can be used, for example, by developers to define markup languages as a means to provide an explicit interpretation of texts independently of devices and systems. It allows for a separation of content and appearance, where the appearance of content encoded in XML can be adopted to different systems using a stylesheet encoded, for instance, in XSLT (Extensible Stylesheet Language Transformation).[2]

XML is used for descriptive markup, where the content and markup are held within the same resource. In the case of XML retrieval, the markup is mainly used to annotate the structure of documents such as paragraphs, sections, etc. Documents have not only content but also structure. The content is the text the document contains while the structure is its logical organisation. XML has become the most widely used standard to encode structured documents.

Thus, XML, contrary to HTML[3], separates the logical structure of documents from the layout. According to the W3C definition of XML, XML documents are ordered, labelled trees. The logical structure of an XML document forms a tree of elements, which starts with a root element and has edges between elements. Content can be found normally in the leaf elements of these XML trees, i.e. those elements that do not have further descendants. The branch elements will contain the structure (e.g. title, paragraph, etc.). The Document Object Model (DOM) is the official tree representation of elements and text. Figure 2.1 is a representation of the XML text below.

```
<article>
    <author>John Smith</author>
    <date>01/01/1970</date>
    <section>
        <paragraph>
```

[1]http://www.w3.org/TR/xml
[2]http://www.w3.org/TR/xslt
[3]http://www.w3.org/html/

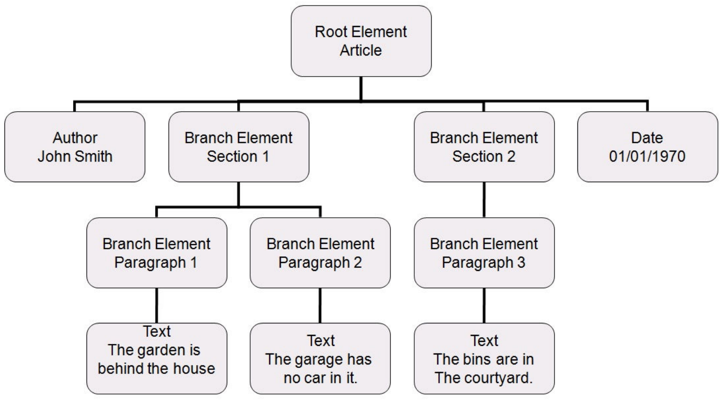


Figure 2.1: DOM example

```
            The garden is behind the house.
        </paragraph>
            The garage has no car in it.
        <paragraph>
        </paragraph>
    </section>
    <section>
        <paragraph>
            The bins are in the courtyard.
        </paragraph>
    </section>
</article>
```

The so-called Document Type Definitions (DTD) define the syntax and vocabulary of an XML document.[1] A DTD contains all possible names of elements and how they can be combined. It defines how elements are related. But, DTD's are written in an non-XML syntax with limited amounts of types that can describe the semantics of elements. Hence, the XML schema[2] language was introduced as another way to define how well-formed XML documents look like. XML schemas are themselves XML documents, which makes them extensible and modifiable in the same way as other XML. An XML schema defines what an allowable XML document is. It constrains the structure. In our example, this could, for instance, mean that paragraphs can only appear as children of sections. We say that an XML element is a child of another XML element if it is embedded in it. In the example above, the paragraph element is a child of the section element. We then also call the section element the parent of the section element. Both DTD and XML schema provide valuable additional information about the structure of a document, which can be used to improve retrieval results.

The final XML-related concept we need to introduce is the XPath.[3] XPath is the non-

[1]http://www.w3.org/TR/REC-html40/sgml/dtd.html
[2]http://www.w3.org/XML/Schema
[3]http://www.w3.org/TR/xpath/

XML language to select XML elements in a document tree. We do not need to go into the details of the XPath syntax for this thesis. It is best explained simply using an example. The XPath expression *article* will select the article element in the example above. The expression *article/author* will find the author, *article//paragraph* will find all paragraphs, while *section/paragraph* will return nothing.

XML schema, tree representation and XPath are concepts that define the theory of XML. In practice, XML is most often used for encoding data [Lalmas, 2009]. This has also been called data-centric XML [Manning et al., 2008], as the emphasis is here on the structural aspects of XML, used to encode complex data values and attributes. Data-centric XML is commonly used for encoding complex structures such as tables in database and their relationships, which often contain non-textual data. Data-centric means regular structure and fairly little content. Examples include attempts to serialize legacy databases in XML as well as data exchanges between databases or towards databases from, for instance, scientific sensors. Both will contain highly structured information, such as particular sensor readings. Data-centric documents are often for consumption by machines, while document encodings are for human consumption. They will contain large amounts of text within XML elements. In XML retrieval, we are interested in the latter. In text-centric XML, XML is a way of treating texts as non-linear structures. From an XML retrieval point of view, texts are 'ordered hierarchies of content objects' (Renear) and their encoding in XML is a reflection of this fundamental principle. We will look into this in more detail in Section 3.3.

In text-centric XML, structure supports the text analysis [Manning et al., 2008]. For text-centric XML retrieval, we can adopt existing IR techniques, as we show in the next section, where we provide a preliminary overview of concepts and challenges in XML retrieval. XML structures help us analyse the information within the text, to improve our relevance ranking of parts of the texts according to an information need. In XML retrieval, we therefore look at XML from a text-centric point of view.

2.3 XML Retrieval

Generally speaking, XML retrieval uses the logical structure of elements and edges between them to aim at returning more precise results to user information needs. It is therefore about retrieving not only relevant document components,[1] but those at the right level of granularity, i.e. those that specifically answer a query. We cover these aims and objectives in more detail throughout this thesis. In this section, we provide a preliminary overview of XML retrieval and relate it to the basic XML retrieval concepts we have just introduced.

The aim of XML retrieval is to use the logical XML-encoded structure of documents to retrieve special parts of documents instead of whole documents [Lalmas, 2009]. In Figure 2.1, if we are looking for where the bin is we expect to retrieve not the whole article but just the particular paragraph. A system should always retrieve the part of a

[1]Please note that we mainly use the term document component as another description of an XML element

document that is most concentrated on an information need. This has also been called the 'structured document retrieval principle' [Manning et al., 2008, p. 201]. Furthermore, we do not just want to retrieve any element that gives the exact answer to our information need but we want those elements that do not contain much else but the answer to our information need. We are looking for the smallest possible element that still fully answers our information need [Lalmas, 2009].

It has proven to be hard to implement systems that address the challenges of granularity in XML retrieval appropriately [Lalmas, 2009]. In order to exemplify some of the underlying challenges, let us have a look again at Figure 2.1. If we have a query looking for the garden behind the house we might want to find only the first paragraph as a result of the retrieval. If we are interested in the whole building, then the whole article gives a better answer. Yet, we would only want the article, if it does not contain too many other paragraphs with irrelevant information.

Therefore, the first major challenge of XML retrieval is to deliver only relevant information and as little irrelevant information as possible. The second challenge is to deliver relevant information only once. This means reducing the overlap in relevance rankings in XML retrieval. Because of the structure of an XML document, if a child is relevant, so will be its parent and further ancestors, as the child is contained by them. They overlap in their information. Overlapping elements that contain each other such as the paragraphs and section in Figure 2.1 are also referred to as 'nested' [Kazai and Lalmas, 2005]. Dealing with these so that a user retrieves all the necessary information but not too much redundant one, is — next to delivering the right level of granularity — the second major challenge for XML retrieval.[1]

For a preliminary insight regarding the implementation issues to address the challenges of returning the best possible XML element (on the right level of granularity and without too much overlap), let us consider as an example the indexing process as part of any XML retrieval system. As we will see throughout this thesis, an enhanced indexing strategy for the documents is one proven way to return different granularities relevant to the user's information need. The XML structure is then exploited before ranking takes place.

An overview of indexing techniques for XML documents is given in [Manning et al., 2008]. The authors collect indexing concepts and implementations of concrete systems. For instance, different parts of an XML document can be indexed with a different value measurement. One approach to enhance the indexing could be to index only non-overlapping elements. If we construct our indexing items so that they do not contain redundant information, we will be able to avoid confronting the user with redundant information. The disadvantage of indexing only non-overlapping elements is that often an XML document cannot be easily divided into good non-overlapping elements [Manning et al., 2008]. Pseudo-elements may need to be introduced. Returning these, however, might not make

[1]Further challenges in XML retrieval are discussed in detail in [Lalmas, 2009] and [Manning et al., 2008]. They include heterogenous XML document collections with many different schemas incorporated in them, or the challenge of building interfaces that help users formulate queries with a correct structure according to a particular set of XML documents. These challenges do not directly influence our thesis, as we concentrate on approaches working with one particular schema and on general retrieval strategies rather than user interface design.

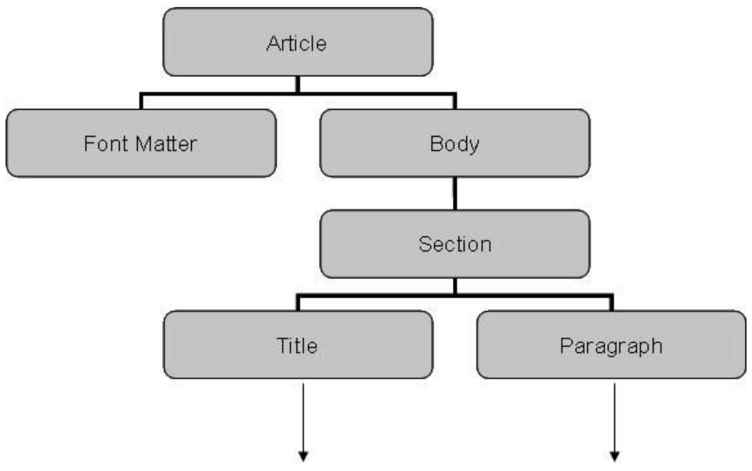


Figure 2.2: Simplified DOM of INEX 2002

sense to the user.

The simplest indexing strategy is therefore often the most preferred one, as we will see in Chapter 5, where we analyse actual XML retrieval systems. Using this simple strategy, all XML elements are considered to be complete information sources in themselves and entered into the index as separate and independent entries. Many XML retrieval systems use this approach and put the burden of finding the most specific and least redundant information therefore fully on the second part of the overall retrieval process, the relevance algorithm.

XML retrieval also requires a different evaluation framework, within which new strategies such as the one discussed for enhanced indexing can be developed, tested and evaluated. In order to offer such a framework, the INitiative for the Evaluation of XML retrieval (INEX) [Gövert et al., 2006] has been founded.[1] INEX is a collaborative effort to create a test environment for XML retrieval. It has delivered test collections with evaluation tasks to find out whether retrieval systems fulfil the specific requirements of XML retrieval [Lalmas, 2009]. The early INEX 2002 collection consisted of about 12,000 articles from IEEE journals. The IEEE journal collection was expanded in 2005. Since 2006 INEX has used the much larger English Wikipedia as a test collection. INEX works with one description of a type of XML document, of which we show a simplified version for 2002 in Figure 2.2. We discuss the test collections in more detail in Section 6.2.

As a collaborative effort to enable evaluation, INEX stands in the tradition of the experimental evaluation initiatives in IR with their robust history such as TREC [Voorhees and Harman, 2005][2] and the ensuing evaluation campaigns such as CLEF[3] and NCTIR[4]. One of the major achievements of INEX has been the definition of new evaluation measures to reflect the requirements of XML retrieval evaluation. Until INEX 2005, the general relevance of an element to an information need was measured as *topical exhaustivity*, which

[1] http://inex.is.informatik.uni-duisburg.de/
[2] http://trec.nist.gov/
[3] http://www.clef-campaign.org/
[4] http://research.nii.ac.jp/ntcir/

reflects the extent to which the information contained in a document component satisfies the information need. More specific to INEX is a second evaluation dimension called *specificity*, which reflects the extent to which an XML element focuses on the information need. Exhaustivity and specificity indicate a system's ability to deliver relevant elements of the right granularity. Since 2005, specificity has become the focus of INEX evaluations [Lalmas, 2009].

INEX [Gövert et al., 2006], in order to consider the additional functionality through structure in XML retrieval, distinguishes between content-only (CO) queries and content-and-structure (CAS) queries. These distinctions emphasize the specificity of XML retrieval, its aim to return parts of documents within an appropriate level of granularity. CO queries describe the standard retrieval task, which searches only for the content conditions ignoring the document's structure. Those queries are necessary, as the users might not know what the structure of documents they are searching for looks like. The system will then decide what document or part of a document it returns. Content-and-structure (CAS) topics include structural constraints as in the Figure 2.1. Here, the users specify their target elements and their context.

Structural constraints in queries are specified in the INEX NEXI standard [Trotman and Sigurbjoernsson, 2004]. NEXI (Narrowed Extended XPath)[Lalmas, 2009] is used in INEX as a standard for XML queries. NEXI shares some elements with XPath but extends it according to the requirements of XML retrieval. It is best explained by looking at an example, as the syntax is very similar to XPath. Let us assume we have the following NEXI query:

```
//article//section[about(.,courtyard)]
```

As in XPath double slashes indicate an arbitrary number of elements. Then, the example query specifies a query for sections about severe weather that are part of articles. The dot in the about clause references the section the clause modifies. The about clause is also called the ranking constraint. The sections are ranked according to their relevance to the information need expressed in the about clause.

NEXI is not full XPath, as the only relationship between nodes in a path is descendant [Trotman and Sigurbjoernsson, 2004]. There is no way to specify the child relationship or other XPath axes. Attributes cannot have descendant nodes so may only be specified at the end of a path.

The distinctions made by INEX and related XML retrieval approaches demonstrate that searching within XML documents — using CO queries or using CAS queries — is more difficult than in unstructured documents as now the structural composition of the documents comes into consideration. We argue that a theoretical evaluation, which we introduce in the next chapter, is particularly useful to deal with the challenges, because it offers flexible means of bringing together content and structure information.

The next chapter now turns to the background of the theoretical evaluation, used to analyse XML retrieval. We first place theoretical evaluation in relation to experimental evaluation before we discuss various approaches to perform theoretical evaluations. Our approach is based on aboutness, which is introduced in Section 3.1.1.3. The framework used to define aboutness is then discussed in Section 3.2.

Chapter 3

Theoretical Evaluation

3.1 Introduction: From Experimental Evaluation to Theoretical Evaluation

An IR system as described in the previous chapter implements an IR model. The two terms are sometimes used interchangeably, but there is nevertheless a clear distinction, which is especially important in the context of IR evaluation [Baeza-Yates and Ribeiro-Neto, 1999]. Just like a climate model is developed to help predict climate behaviour, an IR model helps predict relevant answers to an information need. To this end, an IR model consists of means to generate representations of document and query (its indexing behaviour) and of a ranking function that orders documents with regard to the information need [Baeza-Yates and Ribeiro-Neto, 1999]. An IR model is generally used as an abstract blueprint to implement an actual IR system. The performance of an IR system can, for instance, be measured according to its response time and the space it needs to process its results. For IR, however, more interesting are often not such standard system evaluation measures but performance measure of how precisely an information need is answered and how good the ranking is. Thus, the IR model itself needs to be evaluated and not just its actual implementation as an IR system.

The ranking performance of IR models is evaluated in standardised experimental evaluation procedures. Experimental evaluation in INEX and TREC [Voorhees and Harman, 2005] is used to study the behaviour of IR models [Huibers, 1996]. A typical question in an experimental evaluation is to compare IR models with each other: Model A is tested against model B using a collection C. This test is repeated by manipulating various parameters in A and B or by changing the collection C. For instance, new documents can be added to the collection. After the experiments have been evaluated, a hypothesis is formulated that could explain the experimental results. In order to support the hypothesis, standardised statistical evaluation values such as recall and precision are employed. The hypothesis often concludes with a specification of why and when A performs better than B using precision/recall graphs.

Precision and recall are most commonly used to measure the experimental performance of an IR model [Baeza-Yates and Ribeiro-Neto, 1999]. Precision is the fraction of document retrieval that is relevant to an information need.

$$\frac{|relevantDocuments \cap retrievedDocuments|}{|retrievedDocuments|}$$

Recall, on the other hand, describes the fraction of relevant documents retrieved.

$$\frac{|relevantDocuments \cap retrievedDocuments|}{|relevantDocuments|}$$

Both precision and recall rely on the assumption that every document in a collection is known to be either relevant or non-relevant [Baeza-Yates and Ribeiro-Neto, 1999]. It seems, however, hardly possible that any model would have complete knowledge of all relevant documents for each test query and small test collections can never be fully representative. Precision and recall do therefore entail well-researched issues, which we discuss next.

Problems in experimental evaluation based on precision and recall are good indicators of problems with experimental evaluation in general. Problems with precision and recall include those more directly related to XML retrieval:

- Fundamentally precision and recall distinguish only retrieved and non-retrieved documents. But there might be more classes than that. This is particularly important if we change the unit of retrieval as in XML retrieval. We discuss this in Chapter 4.

- The relevance measure, which is the foundation of precision/recall, does not cover the utility of documents. We will see in Section 7.3.2, how the problem of utility vs. relevance is mitigated in some XML retrieval models. In Section 6.4, we develop user agent models, that directly represent XML retrieval information needs rather than pre-formulated individual queries, as in standard evaluation measures.

Because of issues such as these, standardised test collections and experimental evaluation using precision/recall do not always deliver sufficient information about a model's behaviour, in particular in the context of XML retrieval. There are clear disadvantages to this kind of an experimental evaluation approach, if test collections are incomplete or if precision and recall depend on knowledge about the set of all relevant documents [Huibers, 1996].

This thesis suggests another approach to help with the analysis of XML retrieval models that can complement an experimental evaluation as the one in INEX. We present an evaluation based on describing retrieval as a set of reasoning rules using a theoretical framework that we have developed to specifically analyse XML retrieval model. We deliver a more formal means of comparing system behaviour, with which we are able to go deeper into the details of how particular INEX models achieve their results.

3.1.1 Theoretical Evaluation Approaches

Theoretical evaluation has a long tradition in IR research. [van Rijsbergen, 1989] suggested that an experimental approach to IR should be complemented with a theoretical evaluation to match the increasing complexity of the retrieval task in new areas of IR like multimedia and XML retrieval. More recently the British EPSRC funded Renaissance project[1] has used new theoretical models to explain the complex behaviour in multimedia and XML retrieval, while the long-term INEX participants Piwowarski and Lalmas apply quantum theory to investigate structured document retrieval in some of their recent work [Piwowarski and Lalmas, 2009]. All these approaches expect that a theoretical evaluation can offer new insights into the deeper reasoning behaviour of complex retrieval tasks. In their view, a theoretical evaluation is a complementary means to an experimental one if it helps to clarify the assumptions of retrieval models and if it can identify the characteristics

[1]http://renaissance.dcs.gla.ac.uk/

leading to a particular experimental behaviour. In this thesis, we intend to deliver both in the analysis of XML retrieval models, developed and evaluated within INEX.

We focus on logic-based theoretical evaluation, an idea that goes back to an article of Chiaramella and Chevallet *About Retrieval Models and Logic*, in which logic is regarded as a means to investigate general IR models [Chiaramella and Chevallet, 1992]. Later in this section, we present other approaches and justify our choice of a logic-based theoretical evaluation. These logic-based evaluation approaches to IR are generally founded in Cooper's definition of 'logical relevance' from 1971 for a logical view on relevance [Cooper, 1971, p. 24].

> 'A stored sentence is logically relevant to a representation of an information need if and only if it is a member of some minimal premise set of stored sentences for some component statement of that need.'

In this definition, logical relevance is characterised by the topical bearing of a document on an information need.

In such a logical approach, documents are thought of as sets of sentences. If queries are sentences too, then retrieval is the logical implication of the query by the document [Van Rijsbergen, 1986a]. Thus, van Rijsbergen and others have expressed Cooper's logical relevance in terms of the implication $D \rightarrow Q$ [Van Rijsbergen, 1986b]. In [Cooper, 1971], this means that if the query sentences can be derived by the stored document sentences, then the information need can be satisfied. As an example, consider a query like 'Did Cooper coin his idea of logical relevance in 1971?', then a document containing the sentence 'Cooper coined his idea of logical relevance in 1971' would satisfy it.

However, the representation of an information need and information as truth functional sentences is problematic. Experimentally, the Boolean retrieval models, which are based on $D \rightarrow Q$ [Huibers, 1996], have not performed as well as other standard models such as the vector space model [Sebastiani, 1998].[1] In IR, the implication of a document from a query is never fully given, but documents are only likely answers to information needs. As discussed in Section 2.1, there are two simple reasons for this intrinsic uncertainty. Firstly, the representation of an information in queries and documents is not complete and secondly the information need is subjective to the user's opinions. Thus, van Rijsbergen introduces a 'probable implication'. $D \rightarrow Q$ becomes $P(D \rightarrow Q)$ in a new notation delivered in [Van Rijsbergen, 1986a]. P is a probability function and means that the relevance of a document D for an information need represented in Q is based on 'the extent to which Q might be inferred from D' [Sebastiani, 1998].

[Van Rijsbergen, 1986a] has defined his logical uncertainty principle as follows:

> 'Given any two sentences x and y: a measure of the uncertainty of $y \rightarrow x$ relative to a given data set is determined by the minimal extent to which we have to add information to the data set, to establish the truth of $y \rightarrow x$.'

[1] Paradoxes of the material implication in IR reasoning are analysed in [Sebastiani, 1998].

This principle does not specify where the additional information will come from. It can be derived from the document set or can be established by more generic rule systems such as thesauri.

The logical uncertainty principle is one way to avoid the reduction of using the material implication for discussing relevance behaviour. We would like to go a step further and take up van Rijsbergen's [Van Rijsbergen, 1986a] translation of $D \to Q$ as D 'answers' Q, where the topics in the documents are considered on the most abstract level as answers to the topics formulated in the query. We say that on the most abstract level D is *about* Q.

Following Huibers' formalism and approach [Huibers, 1996], we would like to extend the idea of a topical implication between query and document to the concept of 'aboutness' based on a logic-based framework. Topically speaking, on the most abstract level documents are 'about' queries. With aboutness, we aim to theoretically capture an IR model's behaviour. In this approach, good IR models are those which have a well-defined aboutness relation.

We follow the definition of theoretical evaluation in [Wong et al., 2001], according to which a theoretical evaluation using aboutness

> 'attempted to symbolically characterize qualitative aspects of the matching function, which, up to that point, were normally hidden in the numeric expressions of these functions. In a broad sense an attempt was made to flesh out the assumptions underpinning matching functions.'

As other forms of evaluation in IR, a theoretical evaluation therefore aims to understand the behaviour of IR models. Before we cover our particular usage of logic for theoretical evaluation based on aboutness, we need to consider other theoretical evaluation approaches. We discuss their advantages and disadvantages to justify our decision to use aboutness. There have been many theoretical evaluation approaches in the history of IR, and we only present two more recent examples, one based on a probabilistic framework and another one based on retrieval heuristics.

3.1.1.1 Embedding

Embedding formalises one particular IR model that covers several other models [Huibers, 1996]. For instance, in [Turtle and Croft, 1991] inference networks are used to analyse other models. Inference Networks are known to combine several sources of evidence for the relevance of a document to an information need [Turtle and Croft, 1991]. As such, Inference Networks are instances of the more general idea of using probability theory to analyse IR models. Hence, we concentrate on why embedding in probabilistic models is not general enough to deliver a framework for our analysis. Using aboutness in a logic-based framework, we offer a universal theory for evaluating IR.

Probability theory has frequently been used for a universal framework of IR (for instance in [Nie, 1992]) if the uncertain truth in van Rijsbergen's $P(D \to Q)$ is translated as the probability that given D we can imply Q and the retrieval process as a whole is seen as a probabilistic inference. Thus, faced with the intrinsic uncertainty of the query's and document's inference, we could be tempted to leave our representation as an implication or aboutness and simply write $P(D|Q)$ [Sebastiani, 1998]. However, then we would have already reached a concrete representation of D 'answers' Q in specific IR models, as probability theories have also been used to compute whether a document is relevant for a given query. Using probability for a theoretical evaluation, there will be a bias in favour of probability models such as language models.

We also think that $P(D|Q)$ cannot describe IR models in general, as, e.g., the important principle of transitive reasoning cannot be embedded in probabilistic reasoning (Section 4.4.1). [Sebastiani, 1998] has shown that the assumptions $P(B|A)$ and $P(C|B)$ do not generally lead to $P(C|A)$. Transitivity, however, might be a quality of an IR model, as it is for Boolean retrieval [Huibers, 1996]. Finally, and maybe most importantly, it is not straightforward to embed XML structure in a probabilistic framework.

Therefore embedding in general and embedding in probabilistic frameworks in particular are not general enough to fit our purpose.

3.1.1.2 Retrieval Heuristics

More recent studies using a theoretical evaluation approach are the ones by Hui Fang, Cheng Xiang Zhai and Tao Tao [Fang et al., 2004] [Fang and Zhai, 2005]. They present a formal study and a universal framework for the analysis of IR models, using a set of basic desirable constraints that any reasonable retrieval function should satisfy for good retrieval performance. This is close to our approach but their constraints are based on intuitive heuristics rather than formal logic. They use term frequency weighting, term discrimination weighting, document length normalisation, etc. Fang et al. design experiments to see whether some standard IR models such as language modelling or vector spaces implement these constraints and could show that, if a constraint is not satisfied, it often indicates that the IR model could be improved. They are also able to make direct suggestions for improvements. In summary, they see a tight coupling between the question 'what would be a good retrieval model' and which ones of their constraints would then be satisfied.

At first sight, their approach looks similar to ours, their constraints could be seen as similar to our reasoning rules introduced in Section 4.2. However, they do not imply any reasoning of models with their constraints but rely on the generalisation of intuitions mainly related to TF-IDF (term frequency and inverse document frequency) measures.[1] These include the formalisation of a sensible interaction between TF-IDF: if given a fixed number of occurrences of query terms, a document that has more occurrences of discriminative terms (higher IDF) should achieve a better ranking. Such questions are interesting to anybody working on a new IR model before starting the design process.

Their approach has advantages towards ours. Ours is more abstract and high-level, as we will see. As the authors do not employ a high level of abstraction such as aboutness but remain within the parameters of standard measures to improve retrieval directly, immediate recommendations for further improvements of models can be made. This advantage is, however, also a disadvantage when it comes to the analysis of different and new retrieval

[1] TF-IDF is a weight used to evaluate how important a word is to a document in a collection or corpus.

tasks such as XML retrieval. Here, we do not yet have commonly agreed foundations. TF-IDF, for instance, is subject to discussions in XML retrieval, as it does not deal well with the problem of overlap in XML retrieval [Lalmas, 2009], which we will discuss in Section 5.2.

We would like to remain more abstract than both embedding and retrieval heuristics and therefore use an aboutness-based approach. Furthermore, we will mainly use concepts from logic such as monotonic behaviour. To us, on a very abstract level, a relevance score is a function with various variables that include often terms and their frequency values, but also other parameters. We suggest to study aboutness rules and monotonicity and how they behave with respect to these variables. We will do that extensively in Chapter 5, where we theoretically evaluate retrieval models using aboutness and the topical implications between query and document.

3.1.1.3 Aboutness

Following Huibers' formalism and approach to analyse IR models [Huibers, 1996], we describe how a document topically answers a query using 'aboutness'. Aboutness is described by formally deriving the reasoning process involved in IR models. Huibers has developed a general IR framework based on aboutness that he uses to evaluate existing retrieval models rather than to develop new ones. Such a framework needs to have the formal means to model the underlying concepts and behaviour of any information retrieval model. It should therefore abstract from specific constructs and implementation details.

In this thesis, we present a framework based on aboutness that allows one to analyse the characteristics of particular XML retrieval models. 'Aboutness' has been frequently discussed in IR literature, most notably in the work of [van Rijsbergen and Lalmas, 1996], [Bruza and Huibers, 1994] and [Wong et al., 2001]. Huibers demonstrated the power of an aboutness-based framework for the theoretical evaluation of IR. He successfully derived *aboutness proof systems* to capture several aspects of the reasoning process involved in commonly used flat document retrieval models [Huibers, 1996].

[Wong et al., 2001] present theoretical evaluations of flat document retrieval systems grounded in a well-defined set of steps as a complement to traditional experimental evaluation of IR systems. They found explanations for model behaviour that escaped more traditional evaluation methods, e.g. the effect monotonicity has on aboutness, which we will also discuss in this thesis in more detail. Furthermore, they derived the conditions and thresholds many flat document IR models use to adjust their reasoning behaviour to particular retrieval tasks. We will see in Section 5.2 how this method is also widely used in XML retrieval to deliver more precise answers to an information need. [Wong et al., 2001] also claim that aboutness-based evaluation is more open to debate, as sometimes the underlying assumptions of IR performance can be hidden by tuning a priori assigned parameters in such a way that they fit best the evaluation task. They discuss this for performance differences of the general vector space model compared to the one that uses thresholds. We identify similar strategies to deliver effective XML retrieval models.

We believe that these existing results of an aboutness-based theoretical evaluation in flat document IR indicate that it can also be a powerful methodology to analyse the more complex tasks in XML retrieval. Our thesis is that particularly in the domain of structured document retrieval, aboutness-based theoretical evaluation presents a powerful methodology to analyse the complex interaction of structure and content in XML retrieval.

Aboutness is not equivalent to the more common IR notion of relevance [Manning et al., 2008]. Aboutness captures an abstract topicality relation between document and query. Relevance itself is subjective to a user's perception of the information need. With respect to aboutness, we say that a document can only be relevant to a query if it is also about the query. Yet, a document can be about a query without a user acknowledging its relevance. As a condition of relevance and as being independent of the subjectivity of an information need, aboutness is 'objective'.

Relevance is subjective and aboutness 'objective'. They are not equivalent, as relevance expresses also a user's taste while aboutness is a topical relation between representations of information. With aboutness, we state 'once and for all, what relationship between a document and a request is to hold to compute probable relevance.' [van Rijsbergen, 2004]. We say without aboutness no relevance, but aboutness does not suffice to imply relevance.

Aboutness approaches consider the standard IR techniques such as indexing and matching of query and document as something objective that can be described in a logic-based framework. In the next section, we introduce our logic-based framework using Situation Theory.

3.2 Situation Theory

We use Situation Theory, developed in [Barwise and Perry, 1983], to deliver our aboutness framework for the analysis of XML retrieval. Situation Theory is a mathematical theory of meaning and information with situations as primitives. It offers a logic of information rather than truth assignments and is therefore closer to real-world applications in IR. In the next sections, we are going to explain in more detail why we have chosen Situation Theory.

3.2.1 Basic Concepts

Situation Theory is the mathematical theory of meaning and information and has its roots in a book by Barwise and Perry, meant to present a new science of information [Barwise and Perry, 1983]. Jon Barwise attempted to grasp perceptual messages such as 'Jack saw John running'. The intuition was to find a way to describe that Jack was not only seeing John, but him running. John was doing something within the scope of a situation [Devlin, 1991]. In the language of Situation Theory, situations are structured parts of the world (concrete or abstract) such as the running John individuated by an agent such as Jack.

Compared to other logical systems, Situation Theory does not rely on a full understanding of underlying ontologies but emphasizes strongly the notion of information. It has been developed to support an analysis of the way things in the world can represent and convey information. It starts from the particular challenge that information in any message can never be uniquely determined. There is no single answer to what information is contained in a message like 'Jack saw John running'. It will depend among other things on the receiver and the constraints on the representation of information.

For Situation Theory, information is the starting point not logical 'truth', which is according to Wittgenstein a statement that is true in all possible worlds [Wittgenstein, 1922]. If Situation Theory is used for an aboutness analysis, inference then becomes a way of processing information or 'reasoning', as we call it in this thesis, rather than concluding the truth. The choice of Situation Theory for the analysis of IR processes is thus motivated by the fact that we can use it to describe a document (or XML element) by the information it carries, rather than by which 'truths' logically hold for it [Huibers, 1996], as we will see in Section 3.2.4.

According to [Devlin, 1994], Situation Theory is not a theory of fixed information but a framework for understanding *information flow*, as a way of understanding how agents communicate a message like 'Jack saw John running' across time and space. In our case, we use Situation Theory to understand the information flow in IR models between information source and need expressed in a query. In Section 3.3, we will discuss in more detail the relationship between Situation Theory and IR.

Dretske has developed the idea of information flow in [Dretske, 1981]. Based on his notion of information flow, perception can be regarded as the process by which information is delivered in an analogue form to a cognitive agent for its selective use. Cognition is described by Dretske as the conversion of the information a cognitive agent receives into a digital form, or as a digitisation. A cognitive agent in this process should aggregate three properties. Firstly, it should have the capability of perception. Secondly, it should be able to concentrate on its specific task. Thirdly, it also should have knowledge not only about its system's settings, but also about the environment [Lalmas, 1996]. Cognitive activity is in this sense essentially a digitisation procedure, which cannot be done without loss. Loss minimalisation is the target of a successful digitisation [Lalmas, 1997].

Finally, the idea of information flow emphasises that Situation Theory is based on a relational theory of meaning. There is only an information flow if an object carries information about itself or another object [Lalmas, 1996]. Lalmas gives an example of the world wide web, in which a page A which is linked to by a page B can be thought of as containing information about it. If a page has a link to itself, it contains information about itself. It is the flow that allows us to understand the meaning of what we perceive and to derive additional information [Huibers, 1996].

Situation Theory, as a theory of information flow between situations, offers an ontology that consists of situations and their types, of their relations and basic information captured in so-called infons [Devlin, 1991]. We first define situations, situation types and constraints, before we cover infons in the following section.

3.2.2 Situations

Situations are the primitives in Situation Theory. They describe that asserting something really means to assume that certain situations hold, i.e. that certain constellations are given with describable properties and with relations among these. In particular, content can be represented as a set of situations, as those in which the content is given.

In Situation Theory, various objects are set up by joining situations, permitting complex and abstract ways of classifying situations. Furthermore, a situation can provide information about another situation if their *types* correspond [Devlin, 1994]. A situation type is, for instance, a fire. It is something that might happen in many situations. Cognitive agents use types to classify the world. A fire is happening somewhere, because we see smoke, for instance.

In Situation Theory terminology agents relate types with *constraints* and thus build an information flow between them. These constraints — a term introduced by Barwise — can model any relationship between situations [Lalmas, 1996]. A constraint describes how something can provide information about something else or in our example how smoke can lead to knowledge about fire.

Constraints are defined as binary relations between situation types. If we have a constraint C that links a situation type T_1 to another type T_2, then, given a situation S_1 of type T_1, there is also a situation S_2 of T_2. S_2 may be equal to S_1, or it may become S_1 through some transformation in time and space, or it may just be a completely different situation. IR systems produce these reasonings in their algorithms. If there is a document having information about garden in it, then it will also be about a query asking for house and garden. We use this kind of reasoning in Section 3.1 to define the functional behaviour of IR models.

Constraints and types are the way IR models develop aboutness, or as we call it, the way they reason about aboutness. Generally speaking, for Situation Theory, situations are in the world, while types and constraints are the domain for a reasoning agent such as the human mind or IR models. In IR, documents and queries are situations. Types and constraints are defined by the IR model in the way it relates documents with queries.

Another key concept of Situation Theory is the infon, which we will discuss in the following section. It further specifies situations as collections of items of information. Infons collect the basic facts and 'hold' for a situation. In Situation Theory, an information item is true, because of the situation in which it is embedded in or because of the infons it has.

3.2.3 Infons

Situation Theory allows reasoning, although not all the information a situation carries is known. There is a lot of information in the utterance situation 'Has Jack come home, yet? It looks like it, his bicycle has been moved from the front to the rear.' There is obviously someone talking, possibly two people talking to each other. They both know Jack. Jack has a bicycle. If the receiver of this utterance also knows about Jack's activities, it might also know that Jack has come back from the post office and so on. In Situation Theory, all these information items are described as infons.

In Situation Theory 'infons' are used as the representation of the information an agent perceives [Devlin, 1991]. We say infons hold for a situation. Situations are the context of infons, infons the targets of situations. Reusing [Huibers, 1996]'s formalisation, which is based on [Devlin, 1991] we describe infons as follows:

Definition 1 *An infon is an item $\langle\langle R, a_1, ..., a_n; i\rangle\rangle$ that represents that the relation R holds (if $i = 1$) or does not hold (if $i = 0$) between the objects $a_1, ..., a_n$.*

R is the relationship between the objects. The value i is the polarity of the infon. If the polarity is 1, we call the infon positive; otherwise the infon is called negative. Throughout this thesis, we do not often find the need to discriminate negative and positive infons. For this reason, we will just omit the polarity if the infon is positive. Infons are denoted by Greek letters: ϕ, ψ, etc.

In the example above, the infon describing that the bicycle is back could look like: $\langle\langle being, bike, back; 1\rangle\rangle$, where the polarity 1 corresponds to where the bike is at this moment. $\langle\langle being, bike, front; 0\rangle\rangle$ expresses that the bike is not (anymore) at the front of the house.

The following infons all describe activities of Jack:

$$\langle\langle Walking, Jack, sea; 1\rangle\rangle$$

$$\langle\langle Walking, Jack, land; 1\rangle\rangle$$

$$\langle\langle Walking, Jack, space; 1\rangle\rangle$$

The relationship 'walking' denotes specific relationships between Jack and in this case places where Jack is. These infons all come together as situation types of Jack's activities. Infons can be parametrised in order to capture such situation types[1]:

$$\langle\langle Walking, Jack, p; 1\rangle\rangle$$

The extensions are for p: $\{sea, land, space\}$. If we 'anchor' p with them, we will get from the situation type to the actual situation.

Theoretical evaluation methodologies for information retrieval need formalisms that are powerful enough to characterize the fundamental properties of retrieval models. We use Situation Theory situations to describe documents and queries. Situation Theory is not the only possible choice for a logical framework for a theoretical evaluation in IR (in fact there are many others), but it matches our requirements well.

3.2.4 The Role of Situation Theory in this Thesis

Though Situation Theory does not directly discuss aboutness, some of its ideas are closely related. How documents provide 'answers' to queries, for instance, can be related to the Situation Theory concept of information flow. In their influential book *Information Flow: The Logic of Distributed Systems*, [Barwise and Seligman, 1997] consider information flow in distributed systems. The book presents a general architecture of information carriers

[1]According to standard Situation Theory formalism [Devlin, 1991], we would have to write $\dot{p}$. We choose to simplify this notation and simply write p.

and how they are connected as well as a theory of the information flow related to these connections. The book is based on the work on signals and information in [Dretske, 1981], who in his work looks for reliable correlations so that if object A has a property P_A, object B has property P_B. Then, we can assume that A is P_A carries the information that B is P_B.

Information Flow in Distributed Systems is rich in topics, formalisms and examples. Here, we are particularly interested in the notion of inference for information flow [Barwise and Seligman, 1997, p. 22], which we consider closely related to the idea of describing IR models through the reasoning they involve [Song and Bruza, 2003]. In the book, inference is considered to be key to information, while retrieving information requires inference. Information inference is defined as:

> 'To a person with prior knowledge k, r being F carries the information that s is G, if the person could legitimately infer that s is G from r being F together with k.' [Barwise and Seligman, 1997, p. 22]

This definition makes inference dependent on an agent that is able to infer from knowledge (for instance an IR model). Furthermore, the background knowledge k describes that an IR model must have the capabilities to infer and form relevance decisions. The prior knowledge k is what is realised in an IR model as a result of the indexing and the functional behaviour of the ranking algorithm. For instance, an IR model might index the situation r that Jack is walking with the two infons $\langle\langle Jack\rangle\rangle$ and $\langle\langle walk\rangle\rangle$ F. We can legitimately infer the query expressing an information need about Jack s, represented by the single query term 'Jack' G. Instead of talking about legitimate inference, we talk about aboutness to describe how a document 'answers' a query. We say, the document with the information that Jack is walking is about the information need in the query about Jack.

In our work, we bracket the question whether an IR model agent was right to infer a certain piece of information. We take at face value everything an XML retrieval model produces, and work from there to analyse aboutness behaviour bottom up. We are first and foremost interested in the theoretical evaluation of XML retrieval and not in designing a new and better model for XML retrieval.

We think that Situation Theory concepts offer a good choice for defining aboutness in IR because Situation Theory starts with the way things convey information or in our case the way XML retrieval is done by various models. From there, Situation Theory works upwards to find regularities. In *Modeling Real Reasoning* [Devlin, 2009] asks the question how information arises in real-life-reasoning. He states that we can only find information where we find systematic regularities:

> 'In general, then, information can arise by virtue of systematic regularities in the world. People (and certain animals) learn to recognize those regularities, either consciously or subconsciously, possibly as a result of repeated exposure to them. They may then utilize those regularities in order to obtain information from aspects of their environment.' [Devlin, 2009]

Using Situation Theory, we recognise that such regularities do not need to be consistent in the logical sense of the word but agents in the world such as humans or IR models still reason with whatever partial information they might have. This partialness is not wrong but just the way things are, as all agents are situated. All agents must rely on limited information to reason effectively.

As we are interested in Situation Theory as a framework for the analysis of existing models, we are also not affected by the criticism in [Wong et al., 2001] that Situation Theory as a symbolic theory is too complex for the development of new IR models. [Song and Bruza, 2003] also think that Situation Theory is not useful for the development of IR models, because for them it does not adequately represent human reasoning. [Song and Bruza, 2003] therefore suggest to concentrate on a representational model of semantic memory called Hyperspace Analogue to Language (HAL), which takes into account how humans make inferences. A psychological theory might well be more suited to derive heuristics on how to develop new IR models based on actual human reasoning. We, however, are concerned with the discussion of how XML retrieval models reason that an XML element is about a query. For our purpose, the Situation Theory framework works well.

[Devlin, 1994] claims that for an analysis such as ours, which is trying to describe systematic regularities of real-life phenomena, mathematics needs to be fitted to the data and not the other way around. Situation Theory has been designed to work bottom up, in our case from the real-life IR processes in XML retrieval to the more abstract mathematics. According to [Devlin, 1994], logic becomes narrow-minded if it only concerns itself with attempts to preserve the consistency of a notation system, rather than be open to actual reasoning. If we consider information to be the grounds for reasoning, situations, for instance, can be equal even though we are not able to deliver all information in them. If we had to rely on a purely extensional definition of equality, two document situations would only be equal if they contained the same information. Furthermore, to represent information directly has the intrinsic advantage, that we do not need to worry about its consistency. We only model the information content. This means that we can represent information that might be logically meaningless like the existence of two opposing qualities in the same document. Hamlet's 'to be' or 'not to be' does not lead to confusions.

By employing Situation Theory to describe aboutness, we do not primarily use it to invent new IR models or find a new reasoning, but to evaluate existing ones. A theoretical evaluation needs to be conceived of as a tool to distinguish and complement existing evaluation techniques. XML retrieval aboutness, defined in the next section, is at the heart of this approach.

3.3 Situation Theory Aboutness for XML Retrieval

An aboutness theory derived from Situation Theory offers a formal framework to express the reasoning incorporated in IR models. The framework is general enough not to lead to presumptions about models as the logical implication does, but at same time it is specific enough to discriminate between models. This is done by exploring their reasoning properties, as we will show in Chapter 5. We are confident that we can still call such an abstraction a logical approach, as we follow Wittgenstein's dictum [Wittgenstein, 1922] about logic that logical reasoning expresses decisions for or against an object of interest. In this section, we explore how this decision is made in XML retrieval as a combination of information provided by the content and the structure of XML elements.

In Section 3.3.1, we define how structure is part of aboutness, before in Section 3.3.2 we will develop XML retrieval aboutness.

3.3.1 Structure as part of Aboutness

In a Situation Theory formal representation, we capture topical information in documents with situations as aggregations of infons. In general, we need to define how the information is aggregated for each retrieval approach. We show how this can be done in our analysis of actual retrieval models in Chapter 5.

For XML retrieval, documents are aggregations of *document components* differentiated according to XML element types. Document components do not simply describe smaller documents, but are structured information units and replace documents as the targeted information carrier in XML retrieval. Document components can be small, but what really distinguishes them from documents is that they add structure to the aboutness decision and therefore increase its reasoning complexity. Structures are new properties of documents and add information to the aboutness decision.

In order to specify the nature of this aboutness reasoning using the interaction of structure and content, we now discuss three different structural document paradigms: passage retrieval, hypertext retrieval and XML retrieval. We suggest to embed their reasoning in the structured document retrieval paradigms of an IR model developed in [Chiaramella, 2001]. As seen in Section 3.1.1.1, 'embedding' describes a theoretical evaluation approach in IR [Huibers, 1996] that formalises a model in order to describe other models.

In order to further analyse the influence of structure in passage, hypermedia and XML retrieval, [Chiaramella, 2001] has presented an algorithm to represent particularly XML retrieval by dividing it into a 'fetch' and 'browse' phase. In the fetch phase, a pre-selection of document components takes places, which is narrowed down in the browse phase to retrieve the best document component regarding structural constraints. We would like to extend this paradigm to become a generic mechanism to describe the retrieval of document components. To this end, we divide the reasoning process for structured document retrieval into two analytical phases. In the first phase aboutness is decided, while in the second phase aboutness is specified with the help of structural hints.

With the 'fetch and browse' paradigm, we run 'abstract experiments' on the three structured document retrieval approaches. In the fetch phase, we would like to evaluate a pre-selection mostly based on the general relevance of document components, while in the browse phase we consider the structure to better define aboutness in the retrieval process.

In the next section, we use the fetch and browse paradigm to describe different structured document retrieval approaches. For each of the three approaches we are able to specify the general aboutness relation. We can explain differences in the aboutness behaviour as differences in how structure is considered in the browse phase — whether it is not considered at all as in passage retrieval, whether it is considered as an independent constant as in hypermedia retrieval or whether it is seen as an integral part of the content in a document as in XML retrieval.

Passage retrieval, hypertext retrieval and XML retrieval are all examples of developments in IR [Chiaramella, 2001] that assume that structure can be used to further describe the topicality of a document and therefore improve the determination of aboutness. Here, they are taken as paradigmatic examples of structured document retrieval and analysed in two steps. Firstly, they are mapped on to the model of fetch and browse. Chiaramella's model is used to clearly distinguish structure and content aspects of the retrieval process. Secondly, the aboutness relation of the retrieval paradigm is related to the one of flat document retrieval: If D describes the document and Q the query, then $D \Box\!\!\rightsquigarrow Q$ describes how D answers the query Q. Table 3.1 summarises the results of our findings. We define $\Box\!\!\rightsquigarrow$ more formally in Section 4.3.

3.3.1.1 Passage Retrieval

Passage retrieval [Mittendorf and Schäuble, 1994] is one of the earlier approaches to structured document retrieval. It is based on the assumption that a more focussed discussion of information can be found in the passages of a document rather than the complete document. The targeted document components are passages and the document is seen as a sequence of passages. Passages only contain textual data and form a linear structure to represent aspects of the document. Passages can be of fixed or variable length. The indexing process creates the passage document component and either uses the existing document structure or a fixed number of words for each passage [Mittendorf and Schäuble, 1994].

Most importantly, in passage retrieval passages are not regarded as being topically interlinked. Each passage forms a distinct discourse, each document component is independent. Thus, in passage retrieval structure is only used during indexing and not for retrieval. If we consider the fetch and browse paradigm, for passage retrieval in the fetch phase passages D_i are retrieved and no browsing or focusing of the results takes place. Therefore, passage retrieval is expressed by the aboutness relation: $D_i \Box\!\!\rightsquigarrow Q$ with $D \equiv D_1 \otimes ... \otimes D_n$, where $\otimes$ stands for the composition of document components. The problem with passage retrieval is obviously that structure is not considered in each part of the retrieval process, but only during the indexing. Moreover, passage indexing does not necessarily try to reflect the discourse of a document.

Table 3.1: Structured document retrieval paradigms

Structured Document Retrieval Paradigm	*Nature of aboutness relation*
Passage retrieval	$D_i \Box\!\!\rightsquigarrow Q$ with $D \equiv D_1 \otimes ... \otimes D_n$
Hypermedia retrieval	$D \Box\!\!\rightsquigarrow_{pr} (D \Box\!\!\rightsquigarrow_c Q)$
XML retrieval	$R(D,Q) = F(Q \Box\!\!\rightsquigarrow (D \Box\!\!\rightsquigarrow Q))$

3.3.1.2 Hypermedia Retrieval

Hypermedia retrieval is our second structured document retrieval paradigm. So-called links and hyperdocuments form together a space of document components that are clustered via internal and external hyperlinks [Huibers, 1996]. Hypermedia documents are the basis of the world wide web. The retrieval of such documents uses the additional information of those hyperlinks to confirm the relevance of document components. A hyperstructure does not divide the individual document into smaller components, but clusters documents according to hyperlinks.

Most successful for everyday use was the two step strategy of the original PageRank algorithm [Page et al., 1998]. In a simplified view of PageRank, first a query Q is evaluated against hyperdocuments D using conventional retrieval techniques: $D \Box\!\!\rightsquigarrow_c Q$. This step can be called the fetch phase in the generic fetch and browse algorithm. After the fetch, the browse step will consider the structure of the hyperlinks. The result list of the first step will be sorted in descending order according to their so-called PageRank (pr), which is a value calculated on the basis of the link authority of the page. The pages are displayed in this order.

Overall two different and independent aboutness relations are calculated to determine aboutness: $F[D \Box\!\!\rightsquigarrow_c Q | D \Box\!\!\rightsquigarrow_{pr} Q]$. F is a function representing the complete retrieval process to push the results of the first retrieval stage into the arguments of the second: $D \Box\!\!\rightsquigarrow_{pr} (D \Box\!\!\rightsquigarrow_c Q)$. Aboutness is therefore based firstly on the topical relatedness of documents and query and secondly on the authority of the hyperdocument — a value entirely derived from structure. Hypermedia retrieval with such strategies lacks a combined attempt to use structure and content. Fetch and browse follow two independent aboutness relations. In the case of the original PageRank hypermedia retrieval algorithm, the browse step is even calculated independent of content and before the fetch and authority step.

3.3.1.3 XML Retrieval

Out of these three structural retrieval approaches, only XML retrieval fulfils the full paradigm of fetch and browse by integrating structure and content fully. As seen in Section 2.2, XML specifies the discourse in documents by giving a formal representation of their division into document components. As presented, XML documents form a tree of information by using a recursive definition of document content. The advantage of the hierarchical structure is clearly that many information carriers from texts and websites to multimedia documents are commonly presented in a hierarchical structure. The discourse in most texts is structurally organised in sections, subsections, titles, etc., all of which can be easily represented given the flexibility of XML, by creating corresponding XML elements.

As defined in Section 3.2, we consider documents and queries to be situations with infons representing the collection of information in them. The structure of the XML representation of information allows us to focus the document situation on specific topics in predefined components:

1. XML elements are the atomic information units in XML. We translate this into our Situation Theory framework by stating that each XML element is an XML situation. For hypertext retrieval, on the other hand, there is no need to change the basic information unit, as the scope of the retrieval was still the full document.

2. Two or more such atomic information units can be linked. A link between two XML elements is called an edge. The semantic content of two linked units is never independent. Generally, XML elements have to be parents or children of other XML elements. A second and special case are XML attributes that offer either information about the specific element they are linked to or about the complete document tree. In passage retrieval, on the contrary, passages were informationally independent and therefore did not have relational infons. In hypertext retrieval the information flow was strictly separated in a structure and a content flow.

XML attributes are special in that they are not simply children, but properties of other XML elements [Gövert et al., 2006]. Furthermore, they might be informationally related not just to the XML element they are properties of: e.g., an author attribute might be part of an article element. This does not mean, however, that subelements of this article do not have the same author. Unless otherwise specified they do. This example demonstrates that for attributes at least the information in an XML tree is not just aggregated bottom up or ascending. It depends on how the attribute is propagated [Chiaramella, 2001].

This propagation of an attribute's information can be descending as just demonstrated or ascending, as, e.g., in an edited book where the overall author is the sum of all authors of all book sections. If two different information units have two different authors, then their parent will have both as authors. Chiaramella calls those attributes static which only apply to their specific element [Chiaramella, 2001]. XML element names are examples of such static attributes of structured information units. A title element name only declares its content to be a title. It fully depends on the power of the indexing model whether this kind of distinction is translated into the information units representing the document components. Our Situation Theory framework has to be expressive enough to consider all three structural meanings of attributes.

Clearly attributes are special in so far as they do not aggregate information of their context XML elements. They can make an answer to an information more focussed by providing additional information, but this focus does not necessarily specify information in the surrounding XML elements. Apart from the special case of attributes, the 'natural' information flow between XML elements indicates a hierarchy of information in XML documents. We will discuss this in more detail when we look at hierarchical inclusion for XML retrieval in Section 4.7.1.

According to Chiaramella's definition articles in INEX are informationally 'maximal', as they are exhaustive, while the lowest level paragraphs are 'minimal' and very specific in their information return [Chiaramella, 2001]. For Chiaramella, the aim of XML retrieval is to avoid both maximal and minimal information units as answers to information needs [Chiaramella, 2001]. The maximal unit is the document if not the complete document collection, as the complete document collection can be regarded as one large (virtual) tree of XML elements. A user most satisfied with the complete document can, however, hardly be imagined. At the same time, the average user most likely requires more information than given in just one paragraph. She needs to know more about the context by possibly looking at surrounding paragraphs or by looking at information in other more distant paragraphs. Users have to 'browse' around. Only a combination of fetch and browse gives the best results, and XML retrieval integrates both.

Using this fetch and browse analysis of XML retrieval, we are now able to define XML retrieval aboutness.

3.3.2 XML Retrieval Aboutness

As seen in Chapter 2, with XML retrieval come the new notions of specificity and exhaustivity. Taking into consideration these two evaluation dimensions, both [Nie, 1988] and [Chiaramella, 2001] have suggested to enhance van Rijsbergen's original logical implication for structured document retrieval. [Nie, 1988] extends the implication for XML retrieval as follows:

> 'Given a query Q and a document D, the matching R between D and Q is determined by a function F of the exhaustivity of the document about the query (measured by $D \rightarrow Q$) and the specificity of the document about the query (measured by $Q \rightarrow D$): $R(D,Q) = F[P_K(D \rightarrow Q), P_K^{'}(Q \rightarrow D)]$, where $P_K, P_K^{'}$ are two functions that measure the implications' uncertainty, F is a function combining the two implications and K expresses that these functions are evaluated according to knowledge K (...).'

According to [Nie, 1988], 'exhaustivity refers to the *complete* fulfilment of a query by a document, while specificity refers to the fact that the document fulfills *only* these constraints.' He describes them using material implications. As we have noted in Section 3.1.1, we do not want to limit the interpretation of an IR model to an implication, but remain at the level of aboutness and a higher abstraction. Thus in our Situation Theory framework, Nie's formula becomes $R(D,Q) = F[D \Box\!\!\rightsquigarrow Q, Q \Box\!\!\rightsquigarrow D]$ or $R(D,Q) = F(Q \Box\!\!\rightsquigarrow (D \Box\!\!\rightsquigarrow Q))$. F is a unifying function. In Chapter 6 we will see that INEX has introduced such unifying functions as so-called quantisations.

Just as Nie remains at the level of the implication, [Chiaramella, 2001] similarly uses $\subset$ to describe a two-step fetch and browse algorithm for XML retrieval based on the hierarchy of index expressions [Chiaramella, 2001]. In the first step $D \supset Q$ evaluates exhaustivity and in the second step the most specific information unit is selected based on $Q \supset D$. The selection of a more and more specific unit is continued until the most specific one is found or exhaustivity is violated. This is an interesting approach and shows clearly the need to distinguish the two sides of $\Box\!\!\rightsquigarrow$ in the reasoning of XML retrieval models.

Our aboutness approach points to an informational relation and not to a standard mathematical one such as $\subset$ or $\rightarrow$. Informationally query and document can still be related, although their representations are not subsets of each other or do not imply each other. The additional information to satisfy the information need can come from other sources than the information representation. It can be part, e.g., of the IR system's reasoning to link the topic 'house' always to the topic 'garden'. Then a query about gardens might be satisfied by a document about houses and $D \Box\!\!\rightsquigarrow Q$ though $D \not\subset Q$. Moreover, as our aim is to theoretically evaluate existing XML retrieval models in INEX, we do not want to presuppose a particular reasoning behaviour, as explained in Section 3.1.1.3.

According to [Chiaramella, 2001] the most specific answers in XML retrieval are the result of first fetching the exhaustive answers and afterwards browsing through these answers to narrow down the focus. This assumes that (1) delivering specific answers is the main objective of any XML retrieval approach, and that (2) specificity and exhaustivity judgments are based on the same relevant information. As exhaustivity and specificity are based on the same relevant information, the fetch and browse algorithm indicates that the two values of exhaustivity and specificity are not independent. Only if we can conclude exhaustivity are we able to find out about specificity. We will discuss this in more detail in Chapter 6, where we explore the theoretical evaluation of the two INEX evaluation dimensions.

With the fetch and browse algorithm for XML retrieval and Nie's and Chiramella's re-definition of van Rijsbergen's implication, it becomes clear that the ability to discriminate specificity and exhaustivity during the retrieval process depends on the characteristics of the aboutness relation of each XML retrieval model, as analysed in Chapter 5. Only those models able to distinguish the left and the right hand side of $\Box\!\!\rightsquigarrow$ can make a difference between $D \Box\!\!\rightsquigarrow Q$ for exhaustivity and $Q \Box\!\!\rightsquigarrow D$ for specificity. Let us take the hypothetical example of a retrieval based on an exact match between the topics of query and document, where $\Box\!\!\rightsquigarrow$ would be $\equiv$. Then $D \equiv Q$ would be equivalent to $Q \equiv D$ and exhaustivity and specificity are not distinguished. We see here again one of the major differences between data-centric and text-centric XML retrieval, as analysed in Section 2.2. For data-centric XML retrieval, the distinction between exhaustivity ($D \Box\!\!\rightsquigarrow Q$) and specificity ($Q \Box\!\!\rightsquigarrow D$) would make no sense, as it is based on the exact match between search need representation and returned information unit.

In order to define the XML Situation Theory aboutness relation $\Box\!\!\rightsquigarrow$, we use a subsituation-based aboutness criterion, which we introduce next.

3.3.3 XML Situation Theory Aboutness

Our analysis expresses the relationship of information need and information sources as 'aboutness' and uses the symbol $\Box\!\!\leadsto$ to describe a number of reasoning rules that allow models to conclude aboutness. [Bruza and Huibers, 1996] present a general Situation Theory aboutness criterion. They define it as:

Definition 2 *A situation S is about a situation T if and only if T contains one infon i such that situation S is about infon i.*

This definition of aboutness avoids the problems of using the material implication or subset relationship as in [Nie, 1988] and [Chiaramella, 2001]. Huibers and Bruza relax the implication of aboutness in so far as it does not require that S fully satisfies T. Furthermore i does not need to be contained in S, but only topically implied: $S \Box\!\!\leadsto \{i\}$ does not generally mean $S \supset \{i\}$.

Definition 2 of situation aboutness relies on 'at least one' infon and does just specify the existence of an aboutness relation between query and document [Bruza and Huibers, 1996]: $S \Box\!\!\leadsto T$ if and only if $\exists_{i \in T}[s \Box\!\!\leadsto \{i\}]$. Definition 2, however, does not measure the intensity of aboutness, as van Rijsbergen's logical uncertainty principle does [van Rijsbergen, 2000]. We can extend Huibers' and Bruza's original model and interpret van Rijsbergen's logical uncertainty with the aboutness reasoning. We consider the extent to which we have to change the information to make $\Box\!\!\leadsto$ hold and the extent to which we need change $\Box\!\!\leadsto$ itself. That is why we cannot just talk about an implication between query and document component anymore. A model, which only lacks according to van Rijsbergen's definition information in the training data set but otherwise has all the reasoning capabilities to evaluate a query, has got a better defined aboutness relation than a model lacking some of these capabilities.

Definition 2 works well for flat document retrieval models. It, however, leads to problems if we consider XML retrieval aboutness. The one common infon i could be an infon expressing structure, possibly itself bearing no information useful to a user. In XML retrieval, two XML situations could share the same infons expressing structure, as they share the same document type definition. This does not mean, however, that they are about each other, as the following example demonstrates. Let us assume that two documents both consist of one paragraph embedded in a section. Then, the Situation Theory model of both will have the same infons representing this structure. Furthermore, let us assume that the paragraph in the first document is about dogs, whereas the paragraph in the second document is about cats. Therefore they will not be about each other. Aboutness is a relationship of meaning. Structure in text-centric XML only supports meaning but does not create meaning. Therefore, we need to find another, stricter aboutness criterium in order to discriminate the scales more exactly. To do so, we will use the idea of 'subsituations'.

We use the idea of 'subsituations' instead of simple infons and reformulate Definition 2's aboutness criterion as a subsituation-based one. A subsituation is a situation S_i that is part of another situation S, where we count the situation as a part of itself, i.e. a situation is a subsituation of itself. S_i is as a situation a combination of infons [Huibers, 1996] that has a meaning in itself. For an XML document, this implies XML elements with content and therefore meaning. In the example from Section 2.1, the existence of a paragraph within a section is not a subsituation. Only with the additional information that the paragraph is about 'rain', etc., can we have meaning and a subsituation. In order to conclude that a property like INEX specificity does not apply to a situation S, we just need to show that a situation with that property is not a subsituation of S.

S_i is called a 'strict' subsituation if it is not S: e.g., a situation containing information about houses and garden could have as strict meaningful subsituations one about houses and another one about garden. The situation itself is also a subsituation, but not a strict one. By discriminating *strict* subsituations from *non-strict* subsituations, we are able to differentiate aboutness decisions that demand a completely exhaustive or specific match (non-strict subsituation) from those that only expect a partial match (strict subsituation), which is useful to describe user agent reasoning according to INEX (see Section 6.4).

We say that exhaustivity and specificity are characteristics of a situation, because one of its subsituations makes the situation exhaustive and/or specific. Thus, we assume exhaustivity and specificity to be properties of a situation. We look at evaluation criteria like exhaustivity and specificity from a topical aboutness point of view. We take them as concrete properties of information objects, which are descriptions of the topics in XML elements and query. Van Rijsbergen has a similar idea when discussing standard IR evaluation measures like recall and precision [van Rijsbergen, 2004]. As discussed, he believes that aboutness approaches consider properties of documents.

Based on subsituations and the assumption that exhaustivity and specificity are properties, we can now formulate a new Situation Theory aboutness criterion for XML retrieval, which is based on subsituations:

Definition 3 *Using subsituations, we can define exhaustivity and specificity for XML retrieval:*

1. *Exhaustivity: A situation S is exhaustively about a situation T if and only if T has a subsituation T_i such that situation S is about situation T_i.*
2. *Specificity: A situation S is specifically about a situation T if and only if S has a subsituation S_i such that situation T is about situation S_i.*

With this definition, we bring to an end our discussion of Situation Theory and aboutness in XML retrieval. Using this theoretical framework, we are able to proceed to the actual evaluation of XML retrieval and present in the next chapter our methodology for doing so.

3.4 Conclusion

This chapter has offered the background on the second major component of our approach, on theoretical evaluation. We showed how a theoretical evaluation can help overcome some of the shortcomings of experimental evaluations and deliver more formal means of comparing an IR model's behaviour theoretically, with which we are able to go deeper into the details of how particular models within INEX achieve their results. Situation Theory was introduced as our means to undertake a theoretical evaluation. We have particularly drawn to the work of Huibers in [Huibers, 1996] who uses Situation Theory for an axiomatic framework for IR. The basic ontology of Situation Theory was presented in Section 3.2.2 and related to general IR processes. We use situations to represent information carriers in IR such as documents, XML elements and queries, while so-called infons are good to represent individual information items such as keywords in a query.

We do not believe that Situation Theory is the only possible choice for a logical framework for IR (in fact there are many others), but it matches well our requirements. Situation Theory allows for reasoning on the grounds of incomplete information, as it models information rather than 'truth'. Using Situation Theory, we were finally able to derive a new aboutness criterion for XML retrieval based on subsituations.

Chapter 4

Theoretical Evaluation Methodology for XML Retrieval

4.1 Introduction

In this chapter, we show how to theoretically evaluate XML retrieval models. Our methodology works through four steps. The next section introduces the first three steps that our framework shares with those that analyse flat document retrieval models like the ones in [Huibers, 1996] and [Wong et al., 2001]. In the first step a formalism is delivered to express aboutness symbolically. The second step specifies aboutness by deriving rules of reasoning behaviour, while the third step derives a reflection of aboutness boundaries. Section 4.7 presents the fourth step, which is specific to XML retrieval and one of our additions to the discussion of theoretical evaluation methodologies. The fourth step adds the pure type XML retrieval model, which aims to capture the influence of the XML structure on aboutness reasoning.

In Chapter 5, all these steps of the theoretical evaluation are applied to XML retrieval models, which have been successful in the INEX evaluation. We concentrate on successful models, as we would like to demonstrate that we can show differences in models that are mature in the INEX experimental evaluation.

4.2 Theoretical Evaluation Steps

In this section, we introduce the three theoretical evaluation steps — starting with the basic formalism. They are:

1. A formalism to *translate* the information representation of a particular model into a formal symbolic representation (Section 4.3).
   For this so-called *translation*, we first define what [Huibers, 1996] calls an *aboutness language*. Information items in an XML retrieval model are produced by the indexing process, as an abstraction of the information in documents. We use these information items to translate documents into situations. The translation continues with the definition of equivalence and composition. They define how two situations of a particular aboutness language can be equivalent or composed. Finally, the definition of the more semantically oriented operators of preclusion $\perp$ and containment $\rightarrow$ completes the translation. Preclusion expresses that two situations cannot be combined, as their information content contradicts each other. Containment offers the notion of nested information [Wong et al., 2001]. A situation S contains another situation T if T has only information also found in S.

2. A set of reasoning *rules* to describe the functional behaviour of the XML retrieval aboutness (Section 4.4).
   The definition of the functional behaviour of an aboutness system using its aboutness reasoning in Section 4.4 is the most important step in our theoretical evaluation. We prove whether these reasoning rules are part of an aboutness system and if they are whether there are sufficient rules to cover all aboutness decisions possible within an XML retrieval model. The latter is shown in the completeness proof, which follows

the derivation of the reasoning rules. By comparing the kind of rules a particular system incorporates, we are able to give an overall comparison of the functional behaviour of XML retrieval models.

3. A further investigation of aboutness boundaries for particular retrieval systems called *reflection* (Section 4.5).
   The reflection defines typical non-reasoning related boundary elements of retrieval models.

In Section 4.7, we add another step, which is particular to the analysis of XML retrieval systems. We deliver a comparison of an XML retrieval model's formal characteristics with its flat document equivalent and *pure type XML retrieval*. This qualifies the impact of XML structure on the aboutness behaviour. In a theoretical evaluation approach, IR models can be compared by looking at the different qualities models implement and by studying their reasoning behaviour [Huibers, 1996]. However, we do not only want to compare individual models with other models, but also to measure the impact of XML structure on the reasoning. Most XML retrieval models are based on flat document retrieval models. For each XML retrieval model, we develop the theoretical qualities of its flat equivalent. By comparing each model's reasoning behaviour with the one of pure type XML retrieval and the model's flat equivalent, we are able to measure the distance of the specific model from its flat document retrieval equivalent. The derivation of a pure type XML retrieval in Section 4.7 will enable us to measure this distance.

Translation, derivation of reasoning rules, reflection and comparison with the pure type XML retrieval model contribute to our definition of the aboutness system for a particular model. The following section introduces them all. We begin by defining translation and a set of aboutness languages, which we reuse throughout this thesis.

4.3 Translation

Our Situation Theory formalism needs to be general enough to be applicable to any information retrieval model. In the most general definition, documents and queries can be seen as 'situations' [Huibers et al., 1996b]. In these situations, we formally describe the information representation that results from the indexing. This formal, more abstract representation is seen by [van Rijsbergen, 2004, p. 20] as the distinctive feature of a theoretical evaluation based on aboutness:

> 'In discussing aboutness we come from the very concrete notion that index terms represent properties of documents, which we are making more abstract, whereas with relevance we have a very abstract notion which we make more concrete.'[1]

[1] Van Rijsbergen's book integrates the notion of aboutness and relevance into one theoretical framework based on quantum mechanics. 'So, we need to tackle "aboutness" differently and more abstractly, and our proposal is that properties are modelled as observables by self-adjoint linear operators which when applied to an object (image) produce results with probabilities depending on the geometry of the space within which the objects are represented.' [van Rijsbergen, 2004, p. 20]

Our formalism ‘translates’ the index representation into a general, abstract representation as situations. We call the *translation* the symbolic representation of the model’s handling of information. It uses a function *map* that translates the model’s information items into situations. We will see many ways of applying translations in this thesis.[1]

The translation uses a Situation Theory aboutness language. In this section, we introduce the syntax of aboutness languages and in particular the XML aboutness language. We start by reusing [Huibers, 1996]’s definition of the aboutness language:

Definition 4 *The aboutness language is the smallest subset so that if S and T are expressions in the aboutness language, then so will be the expressions $S \Box\!\!\rightsquigarrow T$, $S \Box\!\!\not\rightsquigarrow T$, $S \equiv T$, $S \not\equiv T$, $S \perp T$, $S \not\perp T$, $S \otimes T$, $S \boxtimes\!\!\rightsquigarrow T$, $S \boxtimes\!\!\not\rightsquigarrow T$, $S \rightarrow T$ and $S \not\rightarrow T$.*

This aboutness language is a simplified version of the one developed in [Huibers, 1996]. We reuse many of his symbols to make comparisons between his and our work easier and demonstrate the continuity of thought we consider ourselves to be part of. Finally, throughout the thesis, we use upper-case letters from the middle of the alphabet such as S or T for situations if we are not talking about queries and document components. In that case we use Q and D. Any descriptors these situations represent like keywords but later on also structured information is symbolised with letters from the beginning of the alphabet like A or B.

According to the aboutness language, we represent the aboutness relation between two situations with the symbol $D \Box\!\!\rightsquigarrow Q$, using the same symbol as in [Huibers, 1996]. If we consider documents and queries to be situations, then $D \Box\!\!\rightsquigarrow Q$ means that the information in D is about the information need expressed in Q. In standard IR models, a document containing ‘garden’ and ‘house’ would be about a query asking for ‘garden’. Query and document would share the term ‘garden’, and most IR models consider a document to be relevant to a query if they overlap in index terms. However, there might also be IR models that are not based on an information overlap in query and document and would not consider D to be about Q.

If we have aboutness, we also need a symbol to express non-aboutness. According to the aboutness language, $D \Box\!\!\not\rightsquigarrow Q$ symbolises that D is not about Q. Most standard IR models do not consider the information ‘garden’ to be about ‘house’, as they are two different terms, even if possibly semantically related.

With $\otimes$, we formalise the composition of situations, e.g. ‘house’ and ‘garden’ can be combined to ‘garden house’. Preclusion, symbolised by $\perp$, expresses that information in situations clashes, as we discuss further in Chapter 5. They cannot be meaningfully combined such as ‘flying birds’ and ‘penguins’. If defined at all, preclusion describes mostly semantic relationships [Wong et al., 2001]. Most models we have investigated have no notion of information clashes. However, all models need to have a definition of situation equivalence. $\equiv$ states that two situations are equivalent, i.e. they contain the same

[1]These definitions of translation can be formally brought together in ‘information fields’ [Bruza and Huibers, 1994] as building blocks for aboutness. Our focus is the use of the formalism in the theoretical evaluation of XML retrieval reasoning processes.

information. $\equiv$ states that situations should be compared according to the meaning they bear not just according to the names we give them such as D or Q. Lastly, containment $\rightarrow$ describes that a situation contains at least the same information as another one has. In Boolean retrieval this corresponds to, e.g., the implication that for any valid expression $x \wedge y$, x is also valid.

As discussed in Section 3.2.4, we use infons to model an IR model's view of the information in documents and queries. We follow [Devlin, 1991] and formalise infons with $\langle\langle ... \rangle\rangle$ brackets, as seen in Section 3.2.3, where we have also seen examples of how infons can formalise information in situations. This section defines the formalism for infons we would like to use in the rest of the thesis to represent information that is returned by the indexing of XML retrieval models. Our formalism is best explained using an example of how to represent a 'bag of keywords' as situations and therefore the most common representation of documents and queries in IR models.

A majority of IR models uses mostly keywords as their descriptors without specifying their relationships. Keywords are easy to index but any knowledge of their relationships is given up. We state that the are described by content infons: If t describes a keyword descriptor from the set of all keywords $\mathcal{T}$, then, generally speaking, its corresponding content infon looks like $\{\, \langle\langle Value, t; j \rangle\rangle \,|t \in \mathcal{T}, j \in \{0, 1\}\}$ or simply $\{\, \langle\langle t \rangle\rangle \,\}$ if the infon has positive polarity. Thus, the keyword descriptor *garden* would be $\langle\langle Value, garden; 1 \rangle\rangle$ or simply $\langle\langle garden \rangle\rangle$. A set of descriptors is a situation: $\{\, \langle\langle t_1 \rangle\rangle \,, ..., \,\langle\langle t_n \rangle\rangle \,\}$. A simple example for a situation as a combination of simple content infons is $\{\, \langle\langle house \rangle\rangle \,, \,\langle\langle garden \rangle\rangle \,\}$. Following [Huibers, 1996], we call the language that only contains content infons the basic infon language. We see that many XML retrieval systems use the simple aboutness language in Chapter 5.

An aboutness language can also contain relations, which we need in order to, for instance, express XML structure. N-ary relationships R between descriptors t_j are themselves infons and are modelled by $\langle\langle R, t_1, ..., t_n \rangle\rangle$. We call these relational infons. As we are interested in XML retrieval, we need to look at the relationship between XML elements that transports information from one XML element to another. As discussed in Section 3.3.1, there are two relations of interest to us in an XML document [Chiaramella, 2001]: the attribute relationship and the parent-child relationship. Thus, $R \in \{Attribute, Parent\}$ (see also [Grossman and Frieder, 2004]).

As XML structure can also be considered to be information [Chiaramella, 2001], in Situation Theory we do not have to change our representation, but can express the parent and attribute structure in XML as infons, too. We further define the resulting XML aboutness language when we look at actual indexing techniques to represent structure in Chapter 5. For now, two further examples indicate how we can represent structure and content as a combination of infons. A simple paragraph about 'garden' can be expressed as $\{\, \langle\langle ElementType, Paragraph, p \rangle\rangle \,, \,\langle\langle Value, garden, p \rangle\rangle \,\}$ using a parameter p. A section with two paragraphs is $\{\, \langle\langle ElementType, Section, s \rangle\rangle \,, \,\langle\langle Parent, s, p_1 \rangle\rangle \,, \,\langle\langle Parent, s, p_2 \rangle\rangle \,,$ $\langle\langle ElementType, Paragraph, p_1 \rangle\rangle \,, \,\langle\langle Value, garden, p_1 \rangle\rangle \,, \,\langle\langle ElementType, Paragraph,$ $p_2 \rangle\rangle \,, \,\langle\langle Value, house, p_2 \rangle\rangle \,\}$. We use the relation *Parent* in order to express that the two paragraphs are the children of the section. We consider details of the XML aboutness language and more complex examples in Section 4.7.2.

Next, we look at the rules that describe the reasoning in XML retrieval models.

4.4 Aboutness Rules

[Huibers, 1996] offers a detailed description of reasoning with aboutness. He introduces the rules, according to which 'intermediate decisions' can be combined. These rules form the centre piece of an aboutness proof system [Huibers, 1996], as they describe its functional behaviour. The aboutness decision is specified by the reasoning rules it incorporates. These can be either fully, partially, or not at all supported. A fully supported rule is one that holds under any circumstances, while a rule that is not supported holds under no circumstances.

Among other things, we add the notion of partially or conditionally supported reasoning, i.e. the one that holds under certain conditions to Huibers' rule system. These conditions constrain an otherwise fully supported rule. For instance, in a standard IR reasoning system, we would expect a document D containing 'house' and 'garden' to be about a query Q about gardens. However, the reasoning system might constrain this aboutness by stating that D must at least have more than one occurrence of the infon garden.

We use a subset of rules given by [Huibers, 1996] and by [Wong et al., 2001] to describe aboutness proof systems. From our experience, this subset gives us an overview of XML retrieval models, though we do not claim that this subset includes all the rules that might be useful for analysing IR reasoning. A number of aboutness properties are discussed in the literature without reaching an agreement on what could constitute a core set of aboutness rules. E.g., Wong et al. do not include Transitivity, but Huibers does. This might be due to the fact that Transitivity cannot be represented in any aboutness framework, as explained in Section 3.1.1.1. Wong et al. compare different theoretical frameworks for their use in the theoretical evaluation of IR model, while Huibers works more in depth but uses Situation Theory as one standard logical framework.

We have used those rules that in our experience best describe the mixture of structure and content typical to XML retrieval. There is an ongoing debate about which of the rules best describe aboutness systems, but all of the cited rule sets are based on the meta-theory of non-monotonic reasoning in [Kraus et al., 1990]. In this theory, a series of non-monotonic reasoning rules are presented that have shown to be a good foundation for the theoretical analysis of XML retrieval systems [Huibers, 1996]. We also see throughout this thesis that support for various kinds of non-monotonic reasoning makes XML retrieval models perform better in experimental evaluations.

We agree with [Huibers, 1996] and [Wong et al., 2001] and consider the careful consideration of monotonicity to be an important feature of IR. This is especially true for XML retrieval, where the task of identifying XML elements at the right level of granularity includes the ability of a reasoning system to revise the existing aboutness decision in favour of more specific answers, as we will discuss in Chapter 5.

Once the framework has been fixed, certain aboutness properties are implied by it. We follow Huibers, as we have also decided to use Situation Theory, but we do not use his complete set of rules, but only a subset, and add rules, which have been introduced by Wong et al. and other authors and which, we believe, are better suited to analyse XML retrieval. We also depart from Huibers' approach by following Wong et. al's inductive evaluation framework. For each evaluation, we do not go through just a minimal set of rules that proves the soundness of the reasoning system, but through all the rules. As [Wong et al., 2001], we are interested in using the reasoning rules to benchmark the functional behaviour of XML retrieval models.

Our analysis is related to the one by Huibers in that we re-use parts of his framework and describe aboutness with Situation Theory. It is, nonetheless, very different from Huibers in that we attempt to look at real existing models. [Huibers, 1996] analyses classes of aboutness systems (such as coordinate retrieval), not those embedded in actual models. We look at individuals, which makes it often more difficult to work out the differences, as in the real world these can be less clear cut.

The following discussion of aboutness rules will first cover basic rules, then combination and containment rules before finally discussing how non-aboutness reasoning can enhance IR models.

4.4.1 Basic Rules

By comparing the rules each model incorporates and the way it does so, we are able to give an overall comparison of the retrieval behaviour. We start with the basic rules, which are often supported by aboutness systems. The first basic rule is Reflexivity:

Reflexivity (Re)

$$S \,\Box\!\!\rightsquigarrow\, S$$

It describes that a situation is about itself. Many retrieval systems consider $\{\, \langle\langle garden \rangle\rangle \,\}$ to be about $\{\, \langle\langle garden \rangle\rangle \,\}$. To exclude empty-set aboutness decisions Reflexivity might be expanded to Singleton Reflexivity, denoted by:

Singleton Reflexivity (SR)

$$\{\phi\} \,\Box\!\!\rightsquigarrow\, \{\phi\}$$

We introduce Singleton Reflexivity, as for some aboutness proof systems pure Reflexivity can lead to logical anomalies like a creation out of nothing. We will see what this means when we come to the discussion of models in Chapter 5.

The next basic rule is Transitivity, which represents the following reasoning: If we use a bar to separate assumptions and conclusions for more complex reasoning rules and $\frac{S}{T}$ means that if S then T, then the Transitivity rule states that if $S \,\Box\!\!\rightsquigarrow\, T$ and $T \,\Box\!\!\rightsquigarrow\, V$ can be assumed, then $S \,\Box\!\!\rightsquigarrow\, V$ is also allowed.

Transitivity (Tr)

$$\frac{S \Box\!\!\rightsquigarrow T, T \Box\!\!\rightsquigarrow U}{S \Box\!\!\rightsquigarrow U}$$

Transitive relations play an important role in IR. For XML retrieval, for instance, as a parent element is about the information in its child element and the child element is about the information in the grandchild element, the parent will also be about the grandchild.

The next basic rule is Symmetry. Here, if one can claim that situation S is about situation T one can also claim the reversal that T is about S. This is the case for many basic aboutness systems if, for instance, a document about garden is about another about garden and houses. Then, the latter will be also about the first.

Symmetry (Sy)

$$\frac{S \Box\!\!\rightsquigarrow T}{T \Box\!\!\rightsquigarrow S}$$

The Set Equivalence rule expresses that two set-equivalent sets have the same aboutness decision. There is a Left Set Equivalence rule and a Right Set Equivalence rule. If two situations are about garden and houses, and we know that one of them is also about a document containing houses and courtyards, then the second one will also be about this document.

Set Equivalence (SE)

$$\frac{S \Box\!\!\rightsquigarrow U, S \equiv T}{T \Box\!\!\rightsquigarrow U} \quad \frac{S \Box\!\!\rightsquigarrow T, T \equiv U}{S \Box\!\!\rightsquigarrow U}$$

With the Euclid rule all basic rules are laid out. It states that if S is about T and also U, T is also about U.

Euclid (Eu)

$$\frac{S \Box\!\!\rightsquigarrow T, S \Box\!\!\rightsquigarrow U}{T \Box\!\!\rightsquigarrow U}$$

Euclid can be part of an aboutness system if a query with houses and garden was about a document with houses and courtyards, and about a second one containing houses, then the document about houses and courtyards is also about the one having information about houses.

4.4.2 Combination Rules

Combination rules [Huibers, 1996] bring together new information from given premises and do not simply exploit what is already in the premises.

An important principle in logical reasoning is monotonicity where given a set of formulas X and a formula α from $X \vdash \alpha$ and $X' \supseteq X$ it can be derived that $X' \vdash \alpha$ [Brown et al., 1992]. Similarly, aboutness can be preserved when unifying situations.

Left Monotonic Union (LMU)

$$\frac{S \,\Box\!\!\rightsquigarrow\, T}{S \otimes U \,\Box\!\!\rightsquigarrow\, T}$$

This rule demonstrates that new information only leads to more conclusions and never reverses existing ones. This is not necessarily desirable: a user does not want to receive 'water melon' if she asks for water. Having reached an information like 'water' does not mean that we should always add new information.

There are also right variants of LMU, called Right Monotonic Union (RMU), where the information is added on the right side of the aboutness relation.

Right Monotonic Union (RMU)

$$\frac{S \,\Box\!\!\rightsquigarrow\, T}{S \,\Box\!\!\rightsquigarrow\, T \otimes U}$$

[Huibers, 1996] and [Wong et al., 2001] both argue that the careful consideration of monotonicity is an important feature of IR. In order to explore the relationship of monotonic unions to known issues in IR, we substitute S with D standing for document situations and T with Q standing for query situations. With Left Monotonic Union (LMU), we can say that if a document D is about a query Q, then so is the composition of D and D'. LMU would look like:

$$\frac{D \,\Box\!\!\rightsquigarrow\, Q}{D \otimes D' \,\Box\!\!\rightsquigarrow\, Q}$$

This substitution makes it obvious that aboutness systems supporting LMU have aboutness decisions which are insensible to document length. In an aboutness model unconditionally supporting LMU, a query containing house is not only about documents with house, but equally valid answers are components with house and garden. We can add document component situations without changing the aboutness decision. Looking at RMU with the same pattern of substitution reveals:

$$\frac{D \,\Box\!\!\rightsquigarrow\, Q}{D \,\Box\!\!\rightsquigarrow\, Q \otimes Q'}$$

For systems supporting RMU, query expansion does not change the aboutness decision. This means that models with RMU can expand the original query and gain a higher recall base while at the same time not losing the set of document components the original query was about. Both document and query length are decisive aspects of aboutness decisions, which underlines the importance of monotonicity [Wong et al., 2001].

Cut is another combination rule. If a model allows for Cut reasoning then we can conclude from knowing that two situations S and T together are about a third U and that S is about T that S without T is also about U. We can 'cut' T off. Let us assume a retrieval system for which $\{\langle\langle garden\rangle\rangle, \langle\langle house\rangle\rangle\}$ and $\{\langle\langle house\rangle\rangle\}$ are together about $\{\langle\langle house\rangle\rangle, \langle\langle garage\rangle\rangle\}$, then $\{\langle\langle garden\rangle\rangle, \langle\langle house\rangle\rangle\}$ is also alone about $\{\langle\langle house\rangle\rangle, \langle\langle garage\rangle\rangle\}$, as it is about $\{\langle\langle house\rangle\rangle\}$.

Cut (CU)

$$\frac{S \otimes T \Box\!\!\rightsquigarrow U, S \Box\!\!\rightsquigarrow T}{S \Box\!\!\rightsquigarrow U}$$

Cut implements the idea that the document information content can be shortened without changing the aboutness decision. Right Weakening allows the query to be cut. A document situation $\{ \langle\langle garden \rangle\rangle, \langle\langle house \rangle\rangle, \langle\langle garage \rangle\rangle \}$ can be about a query $\{ \langle\langle house \rangle\rangle, \langle\langle car \rangle\rangle \}$ and its shorter equivalent $\{ \langle\langle house \rangle\rangle \}$.

Right Weakening (RW)

$$\frac{S \Box\!\!\rightsquigarrow T \otimes U}{S \Box\!\!\rightsquigarrow T}$$

The Mix rule states that if two independent situations are about a third, then their combination is also about the third. It is a variant of Left Monotonic Union. If ‘garages’ are about ‘houses’, and ‘courtyards’ are about ‘houses’, then ‘courtyards’ and ‘garages’ will also be about ‘houses’.

Mix (MX)

$$\frac{S \Box\!\!\rightsquigarrow U, T \Box\!\!\rightsquigarrow U}{S \otimes T \Box\!\!\rightsquigarrow U}$$

The last of the basic rules we would like to discuss is called Context-Free And. If a document is about ‘houses’ and also about ‘garages’, it will be also about the combination of ‘houses’ and ‘garages’. Context-Free And is a variant of Right Monotonic Union.

Context-Free And (C-FA)

$$\frac{S \Box\!\!\rightsquigarrow T, S \Box\!\!\rightsquigarrow U}{S \Box\!\!\rightsquigarrow T \otimes U}$$

Basic and combination rules do not consider context, which might lead to problems. Mix, for instance, allows paradoxical conclusions such as: if Socrates is about being mortal, and dog is about being mortal, then also Socrates, the dog, is about being mortal. We need to be careful about allowing such rules in IR reasoning systems.

Another potential disadvantage of these rules is that aboutness systems incorporating several rules cannot exclude paradoxes as a result of the combined reasoning with these rules. One would be the creation of meaning out of meaningless situations. Say $\emptyset$ stands for the meaningless situation — the one without infons and information. If Reflexivity holds then also for the empty-sets: $\emptyset \Box\!\!\rightsquigarrow \emptyset$. We therefore could prove a creatio ex nihilo, which is to be avoided in any logical set [Huibers, 1996]: We start from $\emptyset \Box\!\!\rightsquigarrow \emptyset$. With LMU, we can derive $\emptyset \otimes T \Box\!\!\rightsquigarrow \emptyset$. Symmetry then delivers $\emptyset \Box\!\!\rightsquigarrow \emptyset \otimes T$. Using LMU again, we have $\emptyset \otimes S \Box\!\!\rightsquigarrow \emptyset \otimes T$. Using Set Equivalence twice, we finally arrive at $S \Box\!\!\rightsquigarrow T$. We should avoid that any meaning $(S \Box\!\!\rightsquigarrow T)$ be created out of no meaning $(\emptyset)$. Aboutness systems, that include LMU, Symmetry and Set Equivalence reasoning, should be careful not to also allow for Reflexivity. In such cases, Reflexivity should be constrained to Singleton Reflexivity, in order to start from something.

Section 4.4.3 rules go beyond simple reasoning, as they implement containment. It can be useful for IR to infer additional information next to a given situation from one of its subsituations. This inference can be either explicit like two different names for the same information, or more deep and implicit like, for instance, the rule that a garden is part of a house. Thus, texts about houses might also be interesting for users investigating garden. Structure is another example of an explicit containment relation between parents and their children. Parent XML elements contain at least the information of their children.

4.4.3 Containment Rules

Within a Situation Theory framework, information containment is a binary relation between two (sub-)situations: Following [Wong et al., 2001], we state that $\rightarrow_s$ means a surface containment, while $\rightarrow_d$ means deep containment. If it is clear within the context of the argument, whether we are speaking about surface or deep containment, we will just use $\rightarrow$. We read $S \rightarrow T$ as situation S contains situation T. Information containment models that information is syntactically or semantically nested. We call a syntactic containment a surface containment. In XML retrieval, surface containment is essential, as it models the information flow between children and parent XML elements. Sections are surface-contained in articles, etc., as they are explicit subelements of articles.

Containment is a relationship between subsituations. As defined in Section 3.3.2, we see any situation S or T to be a composition of its subsituations if $S \equiv S_1 \otimes ... \otimes S_i \otimes ... \otimes S_n$ and $T \equiv T_1 \otimes ... \otimes T_j \otimes ... \otimes T_m$ respectively, where $S_1...S_n$ and $T_1...T_m$ are all possible subsituations in S and T respectively. Then, the containment rule states that: If subsituation S_i of a situation S contains T_j of T, then S is about T.

Containment (C)

$$\frac{S_i \rightarrow T_j}{S \Box\!\!\rightsquigarrow T}$$

Let us assume that $\{\langle\langle garden\rangle\rangle, \langle\langle house\rangle\rangle\}$ contains, according to our aboutness system, the information $\{\langle\langle house\rangle\rangle\}$. Then, the situation $\{\langle\langle garden\rangle\rangle, \langle\langle house\rangle\rangle, \langle\langle garage\rangle\rangle\}$ is about $\{\langle\langle house\rangle\rangle\}$ as well. Please note that if the subsituation is the child of a parent situation and we would have a typical XML retrieval constellation.

In aboutness systems that allow for Absorption reasoning, a situation $\{\langle\langle garden\rangle\rangle,$ $\langle\langle house\rangle\rangle\}$ that contains $\{\langle\langle house\rangle\rangle\}$, is equivalent to its combination with its subsituation.

Absorption (Ab)

$$\frac{S \rightarrow T}{S \otimes T \equiv S}$$

In Absorption, subsituation reasoning does not have to be discriminated from situation reasoning, which means we can leave out the distinction between S and S_i, because only the subsituation appears in the reasoning. Please note that, as according to Section 3.3.2 a situation is also a subsituation of itself, we can simply write S instead of S_i, if there is no need to explicitly distinguish S from S_i in Absorption.

Containment has its own variant of monotonicity. If a situation $\{ \langle\langle garden \rangle\rangle , \langle\langle house \rangle\rangle , \langle\langle garage \rangle\rangle \}$ is about $\{ \langle\langle garden \rangle\rangle , \langle\langle house \rangle\rangle \}$, which contains $\{ \langle\langle house \rangle\rangle \}$, then this situation will also be about $\{ \langle\langle house \rangle\rangle \}$.

Right Containment Monotonicity (RCM)

$$\frac{S \,\Box\!\!\leadsto T, T \rightarrow U}{S \,\Box\!\!\leadsto U}$$

Non-conflict-containment introduces for the first time preclusion in our rules, covered in more depth when we discuss anti-aboutness rules in Section 4.4.4. Preclusion means that information from two situations cannot be meaningfully composed: $S \otimes T \equiv \emptyset$. Non-conflict-containment states that information that is contained in one another cannot preclude it.

Non-conflict-containment (NCC)

$$\frac{S \rightarrow T}{S \not\perp T}$$

On the contrary, Containment Preclusion allows us to state that if a situation S contains another T and T precludes a situation U, then S also precludes U.

Containment Preclusion (CP)

$$\frac{S \rightarrow T, T \perp U}{S \perp U}$$

[Barwise and Etchemendy, 2002] state that information can be partially ordered with respect to containment. According to [Dretske, 1981], all containment relationships are at least reflexive, anti-symmetric and transitive. Chapter 5 demonstrates how some of the specific challenges for XML retrieval are directly linked to these properties and how some XML retrieval models fail to capture containment and therefore fail to offer structural hints to improve the retrieval results.

Non-aboutness rules exploit preclusion further.

4.4.4 Non-aboutness Rules

Non-aboutness stems from the fact that not all situations can be meaningfully combined. As said, we use $\perp$ to denote preclusion. Preclusion is often mutual as in $\{ \langle\langle dog \rangle\rangle \} \perp \{ \langle\langle cat \rangle\rangle \}$ and $\{ \langle\langle cat \rangle\rangle \} \perp \{ \langle\langle dog \rangle\rangle \}$.

Mutual Preclusion (MP)

$$\frac{S \perp T}{T \perp S}$$

Preclusion is key for any theory of information and suggests that something cannot have two contradicting properties at the same time. For the following rule we take the idea of contradicting information a step further and discuss in more detail the idea of anti-aboutness [Huibers, 1996]. We use $S \boxtimes\!\!\rightsquigarrow T$ to state that situation S is in conflict with situation T or S is anti-about T. Anti-aboutness can be a direct consequence of preclusion, but it can also have other causes. If preclusion is given for two subsituations, then the two situations involved are in conflict with each other and meaning is destroyed:

$$\frac{S_i \perp T_j}{S \boxtimes\!\!\rightsquigarrow T}$$

[Huibers, 1996] clearly elaborates why anti-about ($\boxtimes\!\!\rightsquigarrow$) is not equivalent to not-about ($\Box\!\!\not\rightsquigarrow$). Opposition in logics is more than simple negation. As an example we consider two documents, one containing 'ice cream with vanilla and chocolate', the second 'ice cream with vanilla but without chocolate'. A keyword query 'ice cream vanilla' finds both. Support for anti-aboutness reasoning can help in such cases, using a retrieval engine that would understand that 'without' usually states an opposition and should therefore not be retrieved in this case. Anti-aboutness is an attempt to describe exactly those cases, that should not be retrieved [Huibers, 1996]. We agree with Huibers that an IR model should not only be good at determining aboutness, but also be good at distinguishing anti-aboutness relations. For instance, [Widdows, 2003] analyses a theoretically motivated approach to disregard unwanted information from the original query in vector models.

The following example is to show the difference between a situation S that is not about another one T, compared to S being anti-about T [Huibers, 1996]. Let us assume, S_i says that the house has a garden: $\{\langle\langle has, \langle\langle house\rangle\rangle, \langle\langle garden\rangle\rangle ; 1\rangle\rangle\}$. T_j states the opposite: $\{\langle\langle has, \langle\langle house\rangle\rangle, \langle\langle garden\rangle\rangle ; 0\rangle\rangle\}$, while U_k describes a completely different situation $\{\langle\langle has, \langle\langle house\rangle\rangle, \langle\langle garage\rangle\rangle ; 1\rangle\rangle\}$. Now, we can say that with $S_i \perp T_j$, also for their supersituations $S \boxtimes\!\!\rightsquigarrow T$ and $S \Box\!\!\not\rightsquigarrow T$. However, though we can state that $S_i \Box\!\!\not\rightsquigarrow U_k$, we cannot state that $S \boxtimes\!\!\rightsquigarrow U$. We simply do not know enough about the relationship between S and U.

Simple Anti-Aboutness (SAA) in our opinion would make this strong assumption that non-aboutness implies anti-aboutness.

Simple Anti-Aboutness (SAA)

$$\frac{S \Box\!\!\not\rightsquigarrow T}{S \boxtimes\!\!\rightsquigarrow T}$$

Negation Rational (NR) states that non-aboutness is preserved under composition. If a situation is not about 'houses', it is not about 'houses' and 'garden', too.

Negation Rational (NR)

$$\frac{S \Box\!\!\not\rightsquigarrow T}{S \Box\!\!\not\rightsquigarrow T \otimes U}$$

Strict Negation Rational (SNR) goes further and states that if a situation about 'nights' is anti-about 'days' it will also be anti-about 'days' and 'sun'. We have to be careful at this point, as anti-aboutness is strict. There is, e.g., no law that would prevent anybody from using day creams at night. Thus, a situation 'night' is not anti-about 'day creams', though it is anti-about 'day'.

Strict Negation Rational (SNR)

$$\frac{S \boxtimes\!\!\rightsquigarrow T}{S \boxtimes\!\!\rightsquigarrow T \otimes U}$$

Generally speaking, we have to be careful with the use of anti-aboutness rules. An aboutness system is inconsistent if it allows to conclude both $S \Box\!\!\rightsquigarrow T$ and its opposite $S \boxtimes\!\!\rightsquigarrow T$. Let us assume we have a system that implements Right Monotonic Union (RMU) and Strict Negation Rational [Huibers, 1996]. From $S \Box\!\!\rightsquigarrow T$ we can with RMU conclude that $S \Box\!\!\rightsquigarrow T \otimes U$. While with SNR, we can with $S \boxtimes\!\!\rightsquigarrow U$ also say that $S \boxtimes\!\!\rightsquigarrow T \otimes U$. Both conclusions are possible, but they contradict each other. The aboutness system is inconsistent.

Many data-centric XML retrieval models include in their reasoning the Closed World Aboutness Assumption (CWAA) — our last non-aboutness rule. This rule has been shown to promote precision [Wong et al., 2001]. It states that only those situations S that contain another situation T can also be about T. For instance, as the information 'cat' does not include the information 'mouse', 'cat' is also not about 'mouse' according to CWAA.

Closed World Aboutness Assumption (CWAA)

$$\frac{S \not\rightarrow T}{S \Box\!\!\not\rightsquigarrow T}$$

CWAA helps improve precision but it does so potentially at the cost of recall, because it ignores partial matching and other possible information flows, which could establish the aboutness relationship between a document and a query. The potentially negative impact of the Closed World Aboutness Assumption in IR has been well investigated [Van Rijsbergen, 1986a]. IR models using CWAA reasoning are often more appropriate for data-centric XML retrieval [Wong et al., 2001].

4.4.5 Conservative Aboutness

Above, we have discussed the advantages and disadvantages of monotonic behaviour for information aboutness. One of the main disadvantages can be seen in the loss of precision leading potentially to inconsistencies. With query expansion, e.g., we are able to expand a query 'house', which is about 'garden', to a query containing 'house' and 'airplane', but still about 'garden'. We lose precision or possibly even meaning, as the example illuminates. Query expansion is an example for right monotonic behaviour. [Wong et al., 2001] suspect that information retrieval reasoning is at least just conservatively monotonic, it might even be non-monotonic.

[Huibers, 1996] argues that at least in the mind of users IR reasoning is non-monotonic. Users do not accept the loss of precision if their information need about ‘birds’ is answered by a document about ‘bird cages’ rather than one that is focussed on ‘birds’ alone. As discussed in Section 2.3, in XML retrieval finding the most focussed element means eliminating related elements from the ranking. If an XML retrieval aboutness system decides that a parent element, though originally about a query, might be less focussed than a child element about the same query, it can eliminate the parent element from the result list to avoid unnecessary overlap. This means that the decision that the parent element is about the query is reversed. Elimination of overlap in XML retrieval is therefore an example of non-monotonic reasoning.

The next set of rules offers more conservative forms of monotonicity to constrain how information is composed [Wong et al., 2001] and enable non-monotonic reasoning. We say that the composition of information can only produce meaning if it does not violate a condition, or if a preclusion is prevented and meaning therefore preserved. The first two conservative monotonicity rules are conservative variants of left and right monotonic composition, where the added information must pass a condition. For Guarded Left Monotonicity we disallow adding the information ‘flying’ to the information ‘bird’, if the query is about ‘Tweety’.

Guarded Left Monotonicity (GLM)

$$\frac{S \,\Box\!\!\rightsquigarrow T, S \not\perp U}{S \otimes U \,\Box\!\!\rightsquigarrow T}$$

Guarded Right Monotonicity controls query expansion if we disallow adding information about ‘flying’ to the query ‘Tweety’, as $\{\, \langle\langle fly \rangle\rangle \,\} \perp \{\, \langle\langle Tweety \rangle\rangle \,\}$.

Guarded Right Monotonicity (GRM)

$$\frac{S \,\Box\!\!\rightsquigarrow U, T \not\perp U}{S \,\Box\!\!\rightsquigarrow T \otimes U}$$

The last two conservative rules further qualify the answers to queries. This is helpful in order to avoid meaningless compositions. Qualified Left Monotonicity [Wong et al., 2001] allows to exclude document components discussing ‘birds’ and the threats of ‘bird flu’ to a query about ‘Tweety’, as $\{\, \langle\langle bird \rangle\rangle \,, \langle\langle flu \rangle\rangle \,\} \perp \{\, \langle\langle Tweety \rangle\rangle \,\}$. As a cartoon bird, Tweety never catches the bird flu.

Qualified Left Monotonicity (QLM)

$$\frac{S \,\Box\!\!\rightsquigarrow T, T \not\perp U}{S \otimes U \,\Box\!\!\rightsquigarrow T}$$

Qualified Right Monotonicity qualifies query expansion. Here, we are not able to add the information ‘bird flu’ to a query about ‘birds’ without changing the aboutness relation for document components about ‘Tweety’.

Qualified Right Monotonicity (QRM)

$$\frac{S \Box\!\!\rightsquigarrow T, S \not\perp U}{S \Box\!\!\rightsquigarrow T \otimes U}$$

The conservative aboutness rules complete our set of reasoning rules to analyse the functional behaviour of IR and XML retrieval models. The next section introduces the reflection step before we cover pure type XML retrieval.

4.5 Reflection

[Huibers, 1996] developed a theoretical *reflection* as a means to find typical aspects of retrieval models. These typical aspects are shared among all models and are therefore qualitative properties like the reasoning rules that can be used to compare aboutness behaviour. He defined four possible typical aspects of an aboutness system, which we use and extend with XML retrieval specific ones in Section 4.7.6. Firstly, the top document is about any query. Secondly, the top query is the one any document is about. Thirdly, the bottom document is never about any query. Finally, the bottom query is the one no document is about.

We simplify Huibers' notation. Then, let $\mathcal{D}$ be the set of all documents and $\mathcal{Q}$ be the set of all queries:

1. A top document D_j is always about any query Q: $\{D_j | D_j \in \mathcal{D}, D_j \Box\!\!\rightsquigarrow Q\}$.

2. A top query Q_j is one any document D is about: $\{Q_j | Q_j \in \mathcal{Q}, D \Box\!\!\rightsquigarrow Q_j\}$.

3. A bottom document D_j is never about any query Q: $\{D_j | D_j \in \mathcal{D}, D_j \Box\!\!\not\rightsquigarrow Q\}$.

4. A bottom query Q_j is one no document D is ever about: $\{Q_j | Q_j \in \mathcal{Q}, D \Box\!\!\not\rightsquigarrow Q_j\}$.

Reading through these reflections, it becomes obvious that some are alternative statements. E.g., if we find a top document all queries are about, it is impossible that there is a bottom query that will never find any answer in the document collections. We can use this to effectively reduce the number of reflections we have to do, as we will see in Section 4.6, where we introduce the first example of an aboutness analysis including the reflection step.

The reflection step indicates important characteristics of the index representation of a retrieval model, as the aboutness rules show important characteristics of the aboutness decision. In our case, we show in Section 5.18 that it will be enlightening to consider whether the index representation of XML retrieval models can deliver a notion of the element that would be always specific to a query. To deliver this in any case most specific element, is a boundary of the XML retrieval decision.

In the next section, we look at a simple example from the world of flat document retrieval. We use this simple example to illustrate all the traditional steps of a theoretical evaluation — translation, derivation of reasoning rules and reflection. In Section 4.7, we finally introduce the pure type XML retrieval step, which is particular to the analysis of XML retrieval systems.

4.6 Aboutness Systems: Example of the Flat Document Vector Space Model

To illustrate our procedure we will now present the evaluation of a well-known flat document IR model, the simple vector space model [Wong et al., 2001], where the indexing uses a simple bag of keywords approach [Baeza-Yates and Ribeiro-Neto, 1999].

4.6.1 Background

In the plain vector space model, documents in a collection are viewed as vectors in a vector space [Manning et al., 2008], in which there is one axis for each term in the collection. If we represent each document in the collection by the bag of keywords it contains, it can be considered as a point in this vector space and represented as a vector to this point. In fact, such a vector space representation has been used as the foundation of many types of information retrieval operations from calculating relevance rankings to document clustering and has been very successful as a means to represent information [van Rijsbergen, 2004].

In the model, d and q are vectors of weighted or binary index terms t. A term can be a word or any other descriptor for the information the document contains. If the terms are weighted, then these weights are normally based on term frequencies and are values between 0 and 1. If they are unweighted, then the terms will be either 0 if the term appears in a document, or 1, if it does not. However, the weighting scheme is immaterial for our discussion. As we are just discussing an example for our methodology, we only consider unweighted index terms. Weighted index terms can be analysed analogously.

With [Baeza-Yates and Ribeiro-Neto, 1999], let the query vector $\overrightarrow{q}$ be $(u_1, ..., u_m)$ and the document vector $\overrightarrow{d}$ be $(t_1, ..., t_m)$. m is the number of index terms in a collection and the terms are given some canonical ordering so that each term can be found at a particular index in all vectors. The similarity of $\overrightarrow{q}$ and $\overrightarrow{d}$ can be calculated in many ways. In Salton's original model [Salton et al., 1975], the relevance of a document d given a query q is estimated using the cosine of the angle between the two vectors of d and q.

$$rsv(d, q) = \frac{\sum_{i=1}^{m} t_i \times u_i}{\sqrt{\sum_{i=1}^{m} t_i^2 \times \sum_{i=1}^{m} u_i^2}}$$

Since $0 \leq t_i \leq 1$ and $0 \leq u_i \leq 1$, rsv varies between 0 and 1.

Next, we translate the vector representations into situations.

4.6.2 Translation

In the translation part, we develop first the *map* function for the model we are analysing. In this case, we need to express the behaviour of a simple bag of keywords indexing approach.

To this end, we define $\chi(d)$ as the descriptor set that is returned by the indexing process as a representation of document d. In our case, let $\chi(d)$ be a set of index terms, which correspond to the n non-zero entries in the vector for d , while $\chi(q)$ corresponds to non-zero entries for query q. For the translation of the simple vector space we use the basic infon language, as defined on page 42. Index terms are then directly translated into infons and the set of all index terms infons is the document situation.

$$map(\chi(d)) = \{ \langle\langle Value, t; 1\rangle\rangle \, | t \in \chi(d)\}$$

Any document situation S representing a document d is thus the set of value infons of index terms of d. As presented in Section 4.3, a common shortform for such infons is to use simply $\langle\langle t\rangle\rangle$. This translation is similar to the one for vector spaces in [Huibers, 1996]. The translation of a query to a query situation is defined in a similar way.

Next we need to define the operators used in the rules from Section 4.4: equivalence, composition, containment and preclusion. Again, we can reuse what has already been defined in [Huibers, 1996] and [Wong et al., 2001]. In particular, we reuse the algorithm in [Huibers et al., 1996a] for parameter replacement:

Definition 5 *The notation $S^{(x,y)}$ represents the replacement of the parameter x in S with the parameter y. The properties of the parameters exchange are defined as follow: $S^{(w,x)(y,z)} =^{def} (S^{w,x})^{y,z}$.*

Using this notation, we can define the operators according to [Huibers et al., 1996a]:

- Equivalence: Given two situations S and T, $S \equiv T =^{def} (\varphi \in S \Leftrightarrow \varphi \in T)$, where φ is any infon based on all keywords in the document collection. In terms of vectors, this means that the underlying vectors for S and T are identical.

- Composition: Given two situations S and T, $S \otimes T =^{def} (S \cup T)^{(p_1,r_1,\ldots,p_n,r_n)(q_1,s_1,\ldots,q_n,s_n)}$ with p,q and r,s being parameters used in S and T respectively. With respect to vectors, we create a new vector using composition that has a non-zero entry wherever either of the underlying vectors for S and T have a non-zero entry.

- Containment: Given two situations S and T, $S \rightarrow T =^{def} (\varphi \in S \rightarrow \varphi \in T)$, where φ is any infon based on all keywords in the document collection. In the vector representation, where the underlying vector for S has a zero entry the underlying vector for T also has a zero entry and there is at least one non-zero entry that both share.

- Preclusion is not applicable. Preclusion is not applicable, as vectors always have a distance to each other, and vectors into the negative information space are not defined. rsv has to be larger or equal than 0 and smaller or equal than 1. The simple vector space model is therefore not able to express anti-aboutness beyond simple non-aboutness, as we will see later. Simple anti-aboutness would mean that we assume that given a situation S and another situation T, the vectors are perpendicular, or

$S \perp T =^{def} (\varphi \in S \to \varphi \notin T \wedge \varphi \in T \to \varphi \notin S)$. This would be equivalent to $S \Box\!\not\rightsquigarrow T$.

According to the containment definition, any document is surface-contained in any other if and only if it contains only infons from the other document. Deep containment is an addition to the simple vector space model.

Next we discuss the rules, that help us define the behaviour of a model. [Huibers, 1996]'s approach is different from the one presented here, as it is not just some of the rules that are repeated for the analysis of the model but all the rules from Section 4.4. Huibers concentrates on the rules that prove completeness and soundness of the set of reasoning rules that describe the model. As discussed in Section 4.4, the approach presented here is therefore akin to [Wong et al., 2001]'s inductive analysis where all rules are considered to be relevant as functional benchmarks of a model's reasoning behaviour. It is important to understand detailed aspects of the reasoning in terms of conservative monotonicity, anti-aboutness behaviour, etc. In particular, one needs to understand which reasoning rules are not given or only given in certain circumstances, as this reasoning behaviour is highly conclusive for understanding experimental behaviour as outlined in Chapter 8.

4.6.3 Rules

The next step will be to define the aboutness proof system for the simple vector space model. First the *vector space aboutness decision* needs to be defined. According to [Huibers, 1996], given a document d represented by the set of descriptors $\chi(d)$ and a query q represented by $\chi(q)$, d is about q if $rsv(\chi(d), \chi(q)) > 0$.[1] In terms of the vector space model, this implies that the vectors for d and q have at least one entry at the same position. They share at least one index term. Both [Huibers, 1996] and [Wong et al., 2001] have shown that a document is about a query in the simple vector space model if they share information. For [Huibers, 1996]'s Situation Theory framework, this entails the proposition that $rsv(\chi(d), \chi(q)) > 0$ if and only if $\chi(d) \cap \chi(q) \not\equiv \emptyset$. We reuse this proposition in the discussion of the aboutness rules.

For vector space retrieval, we would like to exclude Reflexivity in order to avoid logical anomalities as described in Section 4.4. **Singleton Reflexivity** is then given for vector space retrieval. We have to show that assuming $map(A) \equiv \{\phi\}$ and $map(B) \equiv \{\phi\}$, also $rsv(A, B) > 0$, where A and B are sets of index terms. The latter is the case if there is an index term both part of A and B. We have ϕ as a member of both. Thus $A \cap B \not\equiv \emptyset$, and Singleton Reflexivity is given according to the proposition.

Transitivity does not hold, as the example of $S \equiv \{\langle\langle house\rangle\rangle, \langle\langle garden\rangle\rangle\}$, $T \equiv \{\langle\langle house\rangle\rangle, \langle\langle garage\rangle\rangle\}$ and $U \equiv \{\langle\langle garage\rangle\rangle, \langle\langle car\rangle\rangle\}$ shows. Then $S \Box\!\rightsquigarrow T$ and $T \Box\!\rightsquigarrow U$ but not $S \Box\!\rightsquigarrow U$, as their sets of index terms do not overlap. Thus, Transitivity is not given.

[1]Strictly speaking, this is a different function from the rsv above as the arguments are different but giving it a different name would have made the background less readable. Also in future aboutness discussions, we use rsv for all functions that deliver the retrieval status value.

Symmetry is given. Say, $S \equiv \mathit{map}(A)$ and $T \equiv \mathit{map}(B)$. With the premise $S \Box\!\!\!\rightsquigarrow T$, we want to conclude that $T \Box\!\!\!\rightsquigarrow S$. That is straight-forward, as $\cap$ in $A \cap B$ for $S \Box\!\!\!\rightsquigarrow T$ is commutative. Thus, Symmetry is given.

Set Equivalence is given. Let us assume that $\mathit{map}(A) \equiv \mathit{map}(B)$ and $\mathit{map}(A) \Box\!\!\!\rightsquigarrow \mathit{map}(C)$ are given. We have to show that $\mathit{map}(B) \Box\!\!\!\rightsquigarrow \mathit{map}(C)$ is given. From the premises, we know by the definition of *map* that $A \equiv B$ and $A \cap C \not\equiv \emptyset$, which includes $B \cap C \not\equiv \emptyset$. This proves that Set Equivalence holds.

If **Euclid** would be a property of the aboutness system, from $S \Box\!\!\!\rightsquigarrow T$ and $S \Box\!\!\!\rightsquigarrow U$ we would be able to derive that $T \Box\!\!\!\rightsquigarrow U$. Say, that $S \equiv \mathit{map}(A)$, $T \equiv \mathit{map}(B)$ and $U \equiv \mathit{map}(C)$. Then, $A \cap B \not\equiv \emptyset$ and $A \cap C \not\equiv \emptyset$. However, this does not mean that B and C overlap in information, as the following example demonstrates: Let us assume that $\mathit{map}(A) \equiv \{ \langle\langle \mathit{garden} \rangle\rangle , \langle\langle \mathit{house} \rangle\rangle \}$, $\mathit{map}(B) \equiv \{ \langle\langle \mathit{garden} \rangle\rangle , \langle\langle \mathit{car} \rangle\rangle \}$ and $\mathit{map}(C) \equiv \{ \langle\langle \mathit{house} \rangle\rangle \}$. Then $B \cap C \equiv \emptyset$, and Euclid is not given.

Next, the combination rules are demonstrated. In order to prove that **Left Monotonic Union** holds, we need to find out whether $S \otimes U \Box\!\!\!\rightsquigarrow T$ is given if we know that $S \Box\!\!\!\rightsquigarrow T$. Let us assume that $S \equiv \mathit{map}(A)$, $T \equiv \mathit{map}(B)$ and $S \otimes U \equiv \mathit{map}(C)$. Then, $A \cap B \not\equiv \emptyset$, as S is about T and $C \supseteq A$ by definition of map. With the conclusion that $C \cap B \not\equiv \emptyset$, Left Monotonic Union is given.

For **Right Monotonic Union**, we assume that from $S \Box\!\!\!\rightsquigarrow T$ also $S \Box\!\!\!\rightsquigarrow T \otimes U$. Let us assume that $S \equiv \mathit{map}(A)$, $T \equiv \mathit{map}(B)$ and $T \otimes U \equiv \mathit{map}(C)$. $A \cap B \not\equiv \emptyset$, with S about T. $C \supseteq B$ follows from the definition of map. Therefore $A \cap C \not\equiv \emptyset$, and Right Monotonic Union is given.

Cut would allow us to state $S \Box\!\!\!\rightsquigarrow U$, given that $S \otimes T \Box\!\!\!\rightsquigarrow U$ and $S \Box\!\!\!\rightsquigarrow T$. Let us assume that $S \equiv \mathit{map}(A)$, $T \equiv \mathit{map}(B)$ and $U \equiv \mathit{map}(C)$. Then, $(A \cup B) \cap C \not\equiv \emptyset$ and $A \cap B \not\equiv \emptyset$. Yet, this does not necessarily mean that $A \cap C \not\equiv \emptyset$. Cut is not given.

Right Weakening is also not given. From $\{ \langle\langle \mathit{car} \rangle\rangle \} \Box\!\!\!\rightsquigarrow \{ \langle\langle \mathit{house} \rangle\rangle , \langle\langle \mathit{car} \rangle\rangle \}$, we cannot say $\{ \langle\langle \mathit{car} \rangle\rangle \} \Box\!\!\!\rightsquigarrow \{ \langle\langle \mathit{house} \rangle\rangle \}$. Right Weakening is not given.

Mix is supported if Left Monotonic Union is supported, as it is a special case of LMU with the additional knowledge that the added information is about the query, too. Similarly, **Context-Free And** is supported, as Right Monotonic Union is supported.

Deep containment is not given for our simple vector space model. Thus, Containment, Containment Composition, Absorption, Right Containment Monotonicity, Non-conflict-containment, Closed World Aboutness Assumption and Containment Preclusion are all only supported for surface containment for the model. We defined that a situation S contains a situation T if their underlying descriptor sets A and B share at least one information item. Then, obviously $A \cap B \not\equiv \emptyset$.

Absorption follows from the definitions of composition and containment in *map*. **Right Containment Monotonicity** is given, as Right Monotonic Union is given. As preclusion is not defined for the simple vector space model, **Non-conflict-containment** and **Containment Preclusion** are both not applicable. The **Closed World Assumption** is also not given, because two situations might be in no containment relationship but still share index terms.

Because preclusion is not defined for the simple vector space model, all the other non-aboutness rules using it are not applicable: Mutual Preclusion, Guarded Left Monotonicity, Guarded Right Monotonicity, Qualified Left Monotonicity and Qualified Right Monotonicity. There is no inherent way for the simple vector space model to control or qualify its monotonic behaviour. It cannot support conservative monotonicity.

For the non-aboutness rules, we have already excluded **Mutual Preclusion**. **Simple Anti-Aboutness** is more of a statement than a rule. We state that we consider it to be anti-aboutness, if two situations are not about each other. We can show that Simple Anti-Aboutness is the only way for the simple vector space model to support anti-aboutness.

Negation Rational is clearly not given for the model. With it, from $S \Box\!\not\rightsquigarrow T$ we could conclude that $S \Box\!\not\rightsquigarrow T \otimes U$. With, $\{\langle\langle car\rangle\rangle\} \Box\!\not\rightsquigarrow \{\langle\langle house\rangle\rangle\}$, we can still conclude $\{\langle\langle car\rangle\rangle\} \Box\!\rightsquigarrow \{\langle\langle house\rangle\rangle, \langle\langle car\rangle\rangle\}$. Negation Rational is not given. **Strict Negation Rational** is more of a statement, with which we would like to control the behaviour of systems that support Negation Rational in order to avoid inconsistencies, as shown in Section 4.4.4. As Negation Rational is not given, neither is Strict Negation Rational. Therefore, the only non-aboutness rule that could hold is Simple Anti-Aboutness, if we decide that a non-overlap of information would mean a contradiction in the information. This would be, however, a rather strong assumption, as, e.g., $\langle\langle house\rangle\rangle$ and $\langle\langle garden\rangle\rangle$ do not 'syntactically' have an overlap, but can be informationally related.

Thus, we are not able to control the monotonic behaviour using preclusion or anti-aboutness and other rules of the model. There are many other ways of controlling the monotonic behaviour of an IR model. A commonly used method is to introduce a threshold $\theta > 0$ so that in the equation $rsv(\chi(d), \chi(q)) > \theta$. We call such a vector space model a thresholded vector space model [Wong et al., 2001]. Thresholds are an enhancement to the original model developed by Salton. We now briefly analyse some reasoning changes introduced by such a threshold.

Using the example of this model, we would like to introduce the notion of conditionally supported rules, as presented in [Wong et al., 2001]. This time we only have to investigate those rules that are already supported by the simple vector space model, as we said in Section 4.1 that conditions do not create new aboutness behaviour but constrain existing one.

Singleton Reflexivity is still fully supported by the thresholded vector space model. We have to show that under the premises $map(A) \equiv \{\phi\}$ and $map(B) \equiv \{\phi\}$ then $rsv(A, B) > \theta$. Singleton Reflexivity is fully supported, as $rsv(A, B) = 1$, which has to be larger than θ because it is the maximum rsv.

Similarly, for **Symmetry** if the overlap of information is big enough to guarantee $S \Box\!\rightsquigarrow T$, then it must be also big enough to ensure $T \Box\!\rightsquigarrow S$, as in rsv t_i and u_i are interchangeable without changing the overall rsv. Symmetry is still fully supported.

The last one of the simple rules supported by simple vector space is **Set Equivalence**. It is fully supported by the thresholded vector space model because we have not changed the equivalence relation. No formal proof is necessary.

For the combination rules, things are different. Regarding **Left Monotonic Union**, we can say that $rsv(A, B) > \theta$, as $S \,\Box\!\!\rightsquigarrow T$. The question is whether aboutness is preserved if we extend to $S \otimes U \,\Box\!\!\rightsquigarrow T$. Looking at rsv from page 54, rsv could easily fall below the threshold θ if the impact of the extension is stronger on the denominator $\sqrt{\sum_{i=1}^{n} t_i^2 \times \sum_{i=1}^{n} u_i^2}$ than on the numerator $\sum_{i=1}^{n} t_i \times u_i$. Thus, Left Monotonic Union is now only conditionally given.

Right Monotonic Union is also conditionally supported for the thresholded vector space model according to [Wong et al., 2001]. Both monotonic unions are only conditionally supported, as the respective threshold has to be passed. We have shown the impact of thresholds on particular the combination rules, where we add information here. Having shown the impact of thresholds, we skip the remaining rules from Section 4.4, as we only discuss the plain vector space model an example for our method.

4.6.4 Reflection

In the reflection step, we would like to develop those queries and documents that are part of no aboutness relation and those which are part of every aboutness relation.

First, we would like to show for the simple vector space model that the bottom query is $\{\emptyset\}$ or the empty query. We prove this by showing that no document situation D will ever be about the empty query situation Q. Say, $D \equiv map(A)$ and $Q \equiv map(\emptyset)$. Then, $A \cap \emptyset \equiv \emptyset$ and $\emptyset$ will be part of the bottom query. Next, we need to show that there is no other situation that is part of this bottom query. Say this other situation is a single information item $\{\psi\}$. Then according to Singleton Reflexivity $\{\psi\} \,\Box\!\!\rightsquigarrow \{\psi\}$ and $\{\psi\}$ will not be part of the bottom query. Any other situation S can be constructed from $\{\psi\}$ using Left Monotonic Union. Therefore, only $\{\emptyset\}$ forms the bottom query.

Next, we would like to prove that the bottom document that will never be returned is also $\{\emptyset\}$. The proof is similar to the one for the top query, but this time we assume that $Q \equiv map(A)$ and $D \equiv map(\emptyset)$. Then because of $A \cap \emptyset \equiv \emptyset$, $\emptyset$ will the bottom document. Using Singleton Reflexivity and Right Monotonic Union, we can prove that there is no other document.

One might think, that the top query situation Q is the one that includes all the information in the document collection. Only then it seems to be guaranteed that for any D, $A \cap B \not\equiv \emptyset$ with $Q \equiv map(B)$ and $D \equiv map(A)$. The same applies to the top document D. It is the one that contains all the information in the document collections. This would, however, contradict our earlier observation from Section 4.5 stating that bottom query and top document are complementary, as well as top query and bottom document. According to this observation there cannot be a top query, because we already have a bottom document ($\{\emptyset\}$). And indeed, as we allow empty documents to happen, these would not be returned by queries that include all information from the collection. A similar argument applies to the top document. For the simple vector space model, we do not have top documents or top queries.

We continue the development of our theoretical evaluation methodology by introducing pure type XML retrieval. This is the first inductive evaluation step that we especially introduce to cover XML retrieval.

4.7 Pure Type XML Retrieval

The steps covering translation, aboutness rules and reflection have been developed in [Huibers, 1996] as parts of the theoretical evaluation of any retrieval system. They allow the vertical comparison of retrieval systems with other retrieval systems. In XML retrieval, however, we are not only interested in this vertical comparison, but also in a horizontal one. We are interested in how much a retrieval system uses XML structure to support the aboutness decision. Theoretically, we can measure this by comparing the aboutness behaviour of the XML retrieval model with its 'flat' retrieval model equivalent and what we call pure type XML retrieval.

'Pure types' have been developed by the sociologist Max Weber [Weber, 1997 (1903-1917] and have proven to be useful tools for comparative studies of real-life phenomena not only in sociology, for which they have been conceived originally. Weber speaks about 'Idealtypen' in German, which would literally translate to ideal types. We have chosen to use the alternative English translation of pure types instead, as the term ideal carries the additional meaning of something perfect; a confusion we would like to avoid, as does Weber [Brunn, 2007].

Pure types are according to Weber not to be confused with normative recommendations. They are 'Gedankenbilder' (images of the mind) and not ideals in the English sense of the world. For Weber, pure types emphasise a certain characteristic and are not a generalisation of all possible characteristics. With them, we do not want to develop the one and only prescriptive model for XML retrieval. In this sense, pure types are methodological concepts that develop 'Gedankenbilder' with the specific function to allow us to compare 'real-life' XML retrieval models.

A pure type describes aspects of phenomena, but is not meant to describe perfect things nor all aspects of any one particular case. Rather, it has the purpose to emphasise aspects common to most cases of the observed object. In our case, the emphasis will be on the impact of XML structure on the aboutness behaviour as seen through the INEX evaluation dimensions. Weber's pure types idea [Brunn, 2007] fits to our requirements well, as it is a typological term, which we can use to build a classification to help analyse the impact of XML structure onto reasoning processes in IR.

As noted in Section 3.2.4, we are looking to describe systematic regularities in XML retrieval processes bottom-up using Situation Theory. Pure types are the things that make us recognise XML retrieval processes and are analogous to Devlin's 'possible descriptions' [Devlin, 1994], which in his terms enables us to recognise systematic regularities in the real world. To find such regularities, we need some initial abstract structure, but this abstract structure is neither necessary nor perfect in the sense of 'natural laws' or 'normative assumptions'. It is just something we have in the back of our mind if we think of something such as XML retrieval models.

For each XML retrieval model, we compare its reasoning behaviour with those of other models and look at its consideration of XML structure by determining its qualitative distance to its 'flat' document equivalent and the pure type model. The latter step has not

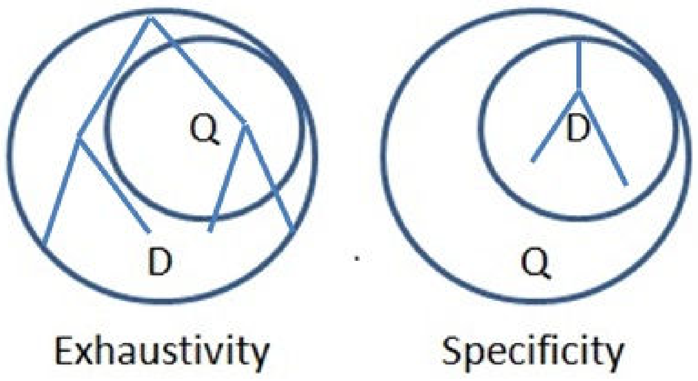


Figure 4.1: INEX View on Exhaustivity and Specificity

been considered yet in aboutness investigations and is particularly useful for the theoretical evaluation of XML retrieval.

To develop pure type XML retrieval, we define its aboutness decision as hierarchical inclusion next.

4.7.1 Hierarchical Inclusion

In this part, we attempt to develop a pure type XML retrieval model, using the two INEX evaluation criteria of exhaustivity and specificity. As discussed in Section 2.3, INEX has two evaluation dimensions:

- Topical exhaustivity reflects the extent to which the information contained in a document component satisfies the information need.
- Component specificity reflects the extent to which a document component focuses on the information need.

As also seen in Section 2.3, exhaustivity describes in how far the document component contains all the information in a query, while specificity describes how little it is about other information than the one in the query. This can be visualised in Figure 4.1. A similar kind of visualisation has been used in [Gövert et al., 2006] to describe the INEX evaluation scales based on exhaustivity and specificity, which we shall analyse in Section 6.3, where we discuss the INEX evaluation scales in more detail.

We use this INEX view on ideal performance in XML retrieval and its visualisation in Figure 4.1 to define pure type aboutness: Say, that a document d is indexed with a descriptor set $\chi(d)$ so that its XML structure is preserved. Then, it will be about a query q (indexed by $\chi(q)$) according to the INEX view if structure and content information of $\chi(q)$ are contained in the structure and information of $\chi(d)$. In Section 4.7.3, we use this relation to define the pure type aboutness operator. According to Figure 4.1, this containment relation defines general relevance or exhaustivity, while specificity is defined as follows: The query XML representation $\chi(q)$ will be about XML document component

$\chi(d)$, if and if only we retrieve only the information demanded by the query and nothing else.

The definition of aboutness for pure type XML retrieval is directly linked to the fact that XML enforces a hierarchical information representation, which is why we call it hierarchical inclusion. XML elements are either totally part of another XML element or not at all. We can map an XML document to an XML tree by defining ancestors and descendants of XML elements with the document being the root of the XML tree: If in a document D a document component $D1$ is contained by component $D2$ then in the corresponding XML tree $D1$ will be a descendant of $D2$. Ancestors can be defined analogously.

Pure type XML retrieval is close to the INEX view of XML retrieval and is still text-centric XML retrieval, as XML structure does not determine aboutness but is a necessary condition of aboutness. To explain this difference, let us assume that we have a section with two paragraphs, one having information about garden and houses, the other having information only about houses. Furthermore, we have a query asking for information about houses in paragraphs. The section is not an answer because the XML structure does not match. Of the two paragraphs the second one is more focussed on the information need. Both paragraphs are a match for the structure specification in the query but only one is more focussed on the sought information. For data-centric XML retrieval, this would be of no interest, as aboutness is decided by XML structure and exact match. For text-centric XML retrieval, however, structure is only a necessary condition, which excludes the section as an answer, but not as an indicator of relevance. For the aboutness decision we need additional reasoning that includes XML structure as a necessary but not sufficient condition of aboutness. Pure type XML retrieval is defined by an aboutness decision that includes content and XML structure equally. We call this aboutness decision hierarchical inclusion.

4.7.2 Translation

For the translation we make the assumption that in pure type XML retrieval the XML structure is preserved during indexing. We want to express the translation of a model that uses XML structure in its matching. To define the translation, we reuse the conceptual graph translation, as defined by Huibers, Ounis and Chevallet in [Huibers et al., 1996a]. Their conceptual graphs map well onto XML trees.

The conceptual graph model has been developed in [Sowa, 1992] and analysed from an aboutness point of view by [Huibers et al., 1996a]. In the model, a query q and a document d are both seen as conceptual graphs. The knowledge in the document collection is modelled as conceptual relationships between concept types. Content is found as referents to concept types. For XML, we consider these concept types to be element types, while we limit the set of relationships to the parent and the attribute relationship. Content is found in XML as part of element types.

In the conceptual graph model from [Huibers et al., 1996a], an XML document d indexed by a conceptual graph $\chi(d)$ is about a query q indexed by the conceptual graph

$\chi(q)$, if the information in $\chi(q)$ is also contained in $\chi(d)$. As previously noted, instead we define that a document d indexed by a descriptor set $\chi(d)$, which preserves the XML structure, is about a query q indexed by $\chi(q)$, if the information in $\chi(q)$ is also contained $\chi(d)$. [Huibers et al., 1996a] use $\leq$ as an aboutness, while we will use $\trianglelefteq$, as we describe later. We further define the aboutness operator in Section 4.7.3 but first we would like to continue the translation with the definition of map.

4.7.2.1 Definition of Map

In this section, we define a translation that preserves the XML structure. We continue to use the model developed in [Huibers et al., 1996a], as it makes the definition of map for pure type XML retrieval straight-forward. As in [Huibers et al., 1996a], we want to represent a graph (in our case an XML tree) as a set of infons in order to preserve the XML structure. Intuitively, we can easily map a hierarchical organisation of information (in an XML tree) to sets, if we consider the parent elements to be the supersets of all sets of information that its children contain. What we need is a way of representing the relationship between parents and children in the same framework. Fortunately for us, we are considering sets of infons, which can either express content or relationships between content in the same formalism.

[Huibers et al., 1996a] develop the approach we are using and state that a conceptual graph carries information and can be seen as a situation. We say that an XML element carries information and can be seen as a situation. For [Huibers et al., 1996a], the conceptual graph situation is constituted of the concepts, referents and relations that define the information the conceptual graph carries. As already seen, we need to consider instead element types, content in element types and parent and attribute relations. As [Huibers et al., 1996a] propose to translate each item of a conceptual graph (concept, referent and relation) into a specific infon, we propose to do the same with XML elements. Using element type, content and relation infons, we next define map for pure type XML retrieval.

An XML tree consists of XML *elements* that have *element types* and associated content, which we refer to as *values*. These elements are connected with edges. We now propose to translate XML elements together with their values into set of infons and to distinguish relational, content and element types.

Let us assume that d is an XML document. Then, it can be translated into situations by using a *map*:

- For each XML element p with element type U in d, map has a result $\{ \langle\langle ElementType, U, p \rangle\rangle \}$, where p is the unique parameter. Such an infon is called an element type infon.

- For each XML element p with a type U containing descriptors k_1 to k_n in d, *map* is $\{ map(U) \otimes \langle\langle Value, k_1, p \rangle\rangle , ..., \langle\langle Value, k_n, p \rangle\rangle \}$. p is the unique parameter that identifies U. k_1 ... k_n is the set of n descriptors (for instance, index terms) that are values of the element type U. We call these infons value infons. $\otimes$ is explained later on.

- Say R is an edge in d and a relationship between two subdocuments A and B of d. Let E_1 and E_2 be element types and $\{\langle\langle ElementType, E_1, p\rangle\rangle\} \in map(A)$ and $\{\langle\langle ElementType, E_2, q\rangle\rangle\} \in map(B)$. p is an identifier for E_1 and q for E_2. We can then say that $map(R(AB)) = map(A) \otimes \{\langle\langle R, p, q\rangle\rangle\} \otimes map(B)$. $\{\langle\langle R, p, q\rangle\rangle\}$ then determines that there is an edge between the elements p and q in d. This can either be a parent edge or an attribute edge. We call such an infon a relational infon. For both types of relations the parameters are ordered so that, for instance, $\{\langle\langle Parent, i_1, i_2\rangle\rangle, \langle\langle ElementType, Article, i_1\rangle\rangle, \langle\langle ElementType, Section, i_2\rangle\rangle\}$ reads as: Article is a parent of Section. Attributes are defined analogously. $\otimes$ is explained later on.

This definition of *map* reuses the one in [Huibers et al., 1996a] for conceptual graphs. XML documents will be represented as sets of infons where each of its XML elements is an element type infon or a combination of value infons with element type infons. These are connected using relational infons representing all edges in an XML document tree.

We further assume a pool of names for parameters that are identifiers for each XML document. Above we use different letters (p and q). In the world of XML, URI's are used to uniquely identify any XML document on the web, while element types are linked to namespaces [Lalmas, 2009]. We could have reused this concept of URI's and namespaces but this would have made our examples very complicated and unreadable. We assume that each XML document is given a unique identifier from an unlimited pool of identifiers.

To translate any XML tree (for an XML document) into a set of infons, we traverse the XML tree in a depth-first manner. Each time we visit an XML element we create an element type infon that contains the type of the element as well as a new parameter from our pool of identifiers. We call the second parameter also the identifier of the element type infon. We make a note that we have visited this element so that the next time we visit the element type we do not create another element type infon. If we reach a leaf we collect all the descriptors in the leaf and create a value infon for each of them. The parameter of the value infon will be the parameter of their element type. We then backtrack though the tree and while backtracking we create relational infons where the first parameter is the identifier of the parent element we are backtracking to, while the second parameter is the identifier of the element type infon we are backtracking from. Following this algorithm, we create a unique XML situation (set of infons) from each XML document.

Furthermore, we can re-create the tree of the XML document bottom up, starting with the value infons to create the leaves and then reconnect the elements by following the relational infons using the element type infons to define the types of the elements. We only allow XML situations (set of XML infons), which lead in such a recreation to a valid XML document according to the definition by the W3C and therefore conform to the rules of a Document Type Definition (DTD) or an XML Schema (XSD) (in our case given by

INEX).[1]

It is customary to create parameters in such a way that their first part identifies the XML tree that is traversed while its second part identifies the XML element in the tree. The example in the next section will demonstrate this.

4.7.2.2 Example

Let us assume we have the following XML document about a garden.

```
<Article>
   <Section>
     <Paragraph>
        The garden is behind that door.
     </Paragraph>
     <Paragraph>
        You arrive at a courtyard with a garden.
     </Paragraph>
    </Section>
</Article>
```

Let us assume that i is the parameter that identifies the whole XML document while i_1 ... i_4 are the parameters to identify individual XML elements. Then, the translation will be the set of infons in Table 4.1.

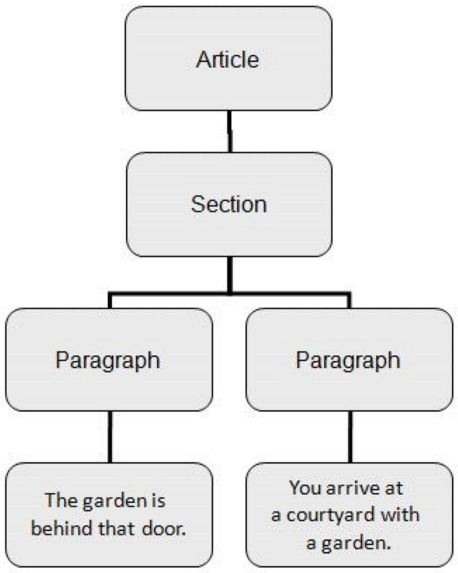

Article
Section
Paragraph
Paragraph
The garden is behind that door.
You arrive at a courtyard with a garden.

Figure 4.2: Pure type translation example

Next, we first present the remaining operators to complete the translation.

[1]This can be proven by running it against the official W3C markup validation service: http://validator.w3.org/

Table 4.1: Pure type translation example

Element Types	$\langle\langle ElementType, Article, i_1; 1\rangle\rangle$ $\langle\langle ElementType, Section, i_2; 1\rangle\rangle$ $\langle\langle ElementType, Paragraph, i_3; 1\rangle\rangle$ $\langle\langle ElementType, Paragraph, i_4; 1\rangle\rangle$
Relational Infons	$\langle\langle Parent, i_1, i_2; 1\rangle\rangle$ $\langle\langle Parent, i_2, i_3; 1\rangle\rangle$ $\langle\langle Parent, i_2, i_4; 1\rangle\rangle$
Values	$\langle\langle Value, door, i_3; 1\rangle\rangle$ $\langle\langle Value, garden, i_3; 1\rangle\rangle$ $\langle\langle Value, arrive, i_4; 1\rangle\rangle$ $\langle\langle Value, garden, i_4; 1\rangle\rangle$ $\langle\langle Value, courtyard, i_4; 1\rangle\rangle$

4.7.2.3 Operators

To perform our aboutness reasoning and to complete the translation, we need to define simple operators between XML situations. These are the last definitions to complete the pure type XML translation. For instance, as seen in Section 4.4, LMU assumes that two situations can be composed. We use four operators in our reasoning rules: equivalence $\equiv$, composition $\otimes$, containment $\rightarrow$ and preclusion $\perp$. We can reuse the definitions of these operators for conceptual graphs in [Huibers et al., 1996a]. In particular, we reuse the algorithm for parameter replacement, as described on page 55.

Let us assume that we have two XML situations S and T, and the translation function map, as defined in Section 4.7.2.1. Then:

- Equivalence: Given two situations S and T with n parameters, equivalence is defined by $S \equiv T =^{def} (\varphi \in (S \cup T)^{(p_1,q_1)\dots(p_n,q_n)}) \Leftrightarrow (\varphi \in T)$ and $(\varphi \in (T \cup S)^{(r_1,s_1)\dots(r_n,s_n)}) \Leftrightarrow (\varphi \in S)$. E.g.: $S \equiv \{\langle\langle Value, house, p; 1\rangle\rangle\}$ and $T \equiv \{\langle\langle Value, house, q; 1\rangle\rangle\}$ are equivalent, because $(\varphi \in (S \cup T)^{(p,q)}) \Leftrightarrow (\varphi \in T)$ and $(\varphi \in (T \cup S)^{(q,p)}) \Leftrightarrow (\varphi \in S)$.

- Composition: We have to consider two composition operators: $\otimes_{rel}$ and $\otimes_{val}$. The first one relates element types and the second one elements types with content. $\otimes_{rel}$: Given two situations S and T, they can be related as $(S \otimes_{rel} R \otimes_{rel} T)$. Parameters in R need to link the element types in S and T: $(S \otimes_{rel} \{\langle\langle R, p, q; 1\rangle\rangle\} \otimes_{rel} T) =^{def} ((S \cup \{\langle\langle R, p, q; 1\rangle\rangle\}^{(s,p)} \cup T)^{(t,q)}$. s and t are parameters in S and T respectively. $\otimes_{val}$: Given a situation S, $(\{Value, v, p; 1\} \otimes_{val} S) =^{def} (\{Value, v, p; 1\} \cup S)^{(p,s)}$, with s being a parameter used to identify an element type infon of S and v a value.

- Containment: Based on the XML structure, we define that containment holds between two element type infons φ and ψ, if the element type of φ is the ancestor of ψ according to an XML schema (e.g., the INEX one). Two situations S and T contain each other if all the element type infons in S contain all element type infons in T. E.g.: Section is a parent of paragraphs in texts. Then: $\{\langle\langle ElementType, Section, p; 1\rangle\rangle\} \rightarrow \{\langle\langle ElementType, Paragraph, q; 1\rangle\rangle\}$, as there is a set of relational parent infons that defines the section to be an ancestor of the paragraph. In this case this is $\{ \langle\langle Parent, p, q; 1 \rangle\rangle \}$.

- Preclusion: We would like to define preclusion as a conflict in the structure of two XML documents according to their XML schemas. This implies that in our case preclusion is not applicable, as we always argue on the basis of the common INEX XML schema. But, generally speaking, two pieces of XML information clash, if their XML schemas are incompatible, either because their element types differ in at least one element type or because these element types are not structured in the same way. For instance, a query looking for information in an XML element B that is the child of an element A cannot find anything useful in a document having B, as the parent of A.

We can define composition using $\cup$, as we express structure with relational infons, which demonstrates the power of the representation, as we can apply set operators for complex information calculations.

Some of these operators like composition allow us to create new situations. We need to ensure that these will always correspond to valid XML documents. We do so by only allowing XML situations that can be recomposed to a valid XML document according to the algorithms described in Section 4.7.2.1. Next, we define formally the aboutness decision as hierarchical inclusion.

4.7.3 Aboutness Decision: Hierarchical Inclusion

As described in Section 4.7.2.1, for pure type XML retrieval, we would like to use the two INEX evaluation measures to define the aboutness decision and use $\trianglelefteq$ to define pure type aboutness. Let us further assume that the descriptor set $\chi(d)$ contains information not only on the content (using index terms) but also on all of the XML structure. We analyse in Chapter 5 various indexing techniques, which realise this. Following the visualisation in Figure 4.1, we define for pure type XML retrieval:

Definition 6 *A document d represented by $\chi(d)$ is about a query q represented by $\chi(q)$ if and only if $\chi(q) \trianglelefteq \chi(d)$, i.e., the information contained in q is also contained in d. This defines exhaustivity, while specificity is defined by $\chi(d) \trianglelefteq \chi(q)$, i.e. d contains only information from q.*

Following [Huibers, 1996], we call the pure type system 'strict' in its aboutness behaviour, as it excludes all elements that do not strictly match the structure requirements expressed in the information need.

In Definition 6, we use structure and content at the same time. In order to make this definition work for XML retrieval, we can rely on our situation theory based representation that includes structure and content values. The following two examples for exhaustivity and specificity aboutness using the pure type map illustrate the definition of hierarchical inclusion aboutness. In the first example, a section containing a paragraph about house and garden is an exhaustive answer to a query asking for a paragraph on house:

Example $\{\langle\langle ElementType, Section, i_1; 1\rangle\rangle, \langle\langle Parent, i_1, i_2; 1\rangle\rangle, \langle\langle ElementType, Para-graph, i_2; 1\rangle\rangle, \langle\langle Value, House, i_2; 1\rangle\rangle, \langle\langle Value, Garden, i_2; 1\rangle\rangle\}$ is exhaustively about $\{\langle\langle ElementType, Paragraph, i_1; 1\rangle\rangle, \langle\langle Value, House, i_1; 1\rangle\rangle\}$.

The paragraph about house and garden is, however, not a particularly specific answer to the same query, as it contains additional unwanted information about garden. With hierarchical inclusion, a paragraph just containing house would be a fully specific answer to a query asking for house, as in the second example:

Example $\{\langle\langle ElementType, Section, i_1; 1\rangle\rangle, \langle\langle Parent, i_1, i_2; 1\rangle\rangle, \langle\langle ElementType, Para-graph, i_2; 1\rangle\rangle, \langle\langle Value, House, i_2; 1\rangle\rangle\}$ is specifically about $\{\langle\langle ElementType, Paragraph, i_1; 1\rangle\rangle, \langle\langle Value, House, i_1; 1\rangle\rangle\}$.

The aboutness system based on hierarchical inclusion describes the properties of an XML retrieval model that only considers elements that are fully contained in each other to be about each other. The two examples directly correspond to the INEX view on exhaustivity and specificity, as visualised in Figure 4.1.

It is important to note that pure type aboutness (though using INEX evaluation dimensions to define its aboutness decision), is not 'ideal', as it, for instance, must ignore elements that are near misses. Near misses have been extensively discussed in INEX and are those elements that might still be seen as relevant to an information need but are not included in the ranking, as they fall through filters to increase specificity (see Chapter 7).

We conclude our discussion of hierarchical inclusion in XML-based aboutness by looking at functional properties hierarchical inclusion supports. Next, we use the definitions of the operators from Section 4.7.2.3 to introduce the reasoning rules an aboutness system based on the defined hierarchical inclusion $\trianglelefteq$ supports. These rules will later on be used to estimate the impact of structure on the reasoning behaviour in XML retrieval.

4.7.4 Aboutness Rules

This section develops the reasoning properties of pure type XML retrieval. First, we introduce a proposition that greatly simplifies the proofs we have to do in the analysis of the pure type model. The proposition is based on our definitions for translation and aboutness decision and Huibers' analysis of conceptual graphs in [Huibers, 1996] and [Huibers et al., 1996a]:

Proposition 4.7.1 *For XML documents A and B, $B \trianglelefteq A$ if and only*
$map(A) \supseteq map(B)$

Proof $\Rightarrow$: Let $B \trianglelefteq A$. Then, we know that B has the content and the structure of a subdocument of A. Using the algorithm from page 64, we construct two situations $S \equiv map(A)$ and $T \equiv map(B)$. This means all relational, element types and value infons from T are also in S. Thus, $map(A) \supseteq map(B)$ with parameter renaming.
$\Leftarrow$: Let $map(A) \supseteq map(B)$. Then, we know that $map(B)$ has only infons also found in $map(A)$. If we apply the algorithm to transform situations into XML documents from

64, B needs to have the structure and content of a subdocument of A and is therefore hierarchically included in it: $B \trianglelefteq A$. $\blacksquare$

The aim of our translation was to use the power of set theory in the aboutness proof. Proposition 4.7.1 verifies that we can do aboutness proofs with a relatively simple set operation on the situation representation of XML trees. The proof uses the proposition proof for conceptual graphs in [Huibers, 1996]. In our case, however, proposition 4.7.1 is also a straight-forward conclusion from the INEX view on XML retrieval aboutness, as visualised in Figure 4.1, and the corresponding idea of hierarchical inclusion.

Using Proposition 4.7.1, we can now easily show that **Reflexivity** is given, as $map(A) \supseteq map(A)$ with $S \equiv map(A)$. **Transitivity** holds, too. If we have $S \Box\!\!\leadsto T$ and $T \Box\!\!\leadsto U$, then also $S \Box\!\!\leadsto U$. Say, that $S \equiv map(A)$, $T \equiv map(B)$ and $U \equiv map(C)$. Then, $map(A) \supseteq map(B)$ as well as $map(B) \supseteq map(C)$. Thus, $map(A) \supseteq map(C)$.

Transitivity holds, but **Symmetry** does not. From $S \Box\!\!\leadsto T$, we do not derive that also $T \Box\!\!\leadsto S$. Again, $S \equiv map(A)$ and $T \equiv map(B)$. Then, $map(A) \supseteq map(B)$ is not equivalent to $map(B) \supseteq map(A)$. The aboutness system is not symmetric.

Euclid is also not supported, as the following example demonstrates: Say S would be $\{ \langle\langle ElementType, Section, i_1; 1\rangle\rangle, \langle\langle Parent, i_1, i_2; 1\rangle\rangle, \langle\langle ElementType, Paragraph, i_2; 1\rangle\rangle, \langle\langle Value, house, i_2; 1\rangle\rangle, \langle\langle Value, garden, i_2; 1\rangle\rangle \}$, T would be $\{ \langle\langle ElementType, Section, i_1; 1\rangle\rangle, \langle\langle Parent, i_1, i_2; 1\rangle\rangle, \langle\langle ElementType, Paragraph, i_2; 1\rangle\rangle, \langle\langle Value, house, i_2; 1\rangle\rangle \}$, and U would be $\{ \langle\langle ElementType, Section, i_1; 1\rangle\rangle, \langle\langle Parent, i_1, i_2; 1\rangle\rangle, \langle\langle ElementType, Paragraph, i_2; 1\rangle\rangle, \langle\langle Value, garden, i_2; 1\rangle\rangle \}$. Then $S \Box\!\!\leadsto T$ and $S \Box\!\!\leadsto U$, but not $T \Box\!\!\leadsto U$. Please note that we try to avoid using examples for our reasoning proofs in the rest of the thesis, as they will easily get very complicated if we need to include XML structure in them.

The basic rule left-over is **Set Equivalence**. Both Left and Right Set Equivalence hold. We prove only Left Set Equivalence, as Right Set Equivalence is the mirror case. Let us assume that $S \equiv map(A)$, $T \equiv map(B)$ and $U \equiv map(C)$. Then, $map(A) \equiv map(B)$ and $map(A) \supseteq map(C)$. From these we have $map(B) \supseteq map(C)$. With $map(B) \supseteq map(C)$, $C \trianglelefteq B$ also holds. Set Equivalence is given, again with no different behaviour for exhaustivity and specificity. Substituting the document component situation or the query situation does not change the behaviour in terms of the INEX evaluation dimensions.

Regarding the combination rules, **Left Monotonic Union** holds for pure type XML retrieval. Given the assumption that any situation S is about another situation T ($S \Box\!\!\leadsto T$), LMU offers the conclusion that also $S \otimes U \Box\!\!\leadsto T$. For the proof, let us assume, that $S \equiv map(A)$, $T \equiv map(B)$ and $S \otimes U \equiv map(C)$. Then, $map(A) \supseteq map(B)$ according to Proposition 4.7.1. With the definitions of map, we also have $map(C) \supseteq map(A)$. Therefore: $map(C) \supseteq map(B)$. Thus, according to the proposition: $B \trianglelefteq C$ and LMU is fully supported. Please note that we do not need to distinguish $\otimes_{rel}$ and $\otimes_{val}$, as the original part of the document that was about the query and that is described by S does not change, whether we add new XML elements or content. Thus, always $map(C) \supseteq map(A)$. Similar arguments apply for the other reasoning rules, which is why we only speak of $\otimes$ for them.

As LMU holds, exhaustivity aboutness is sustained when moving up the XML document tree. Let us assume that we know that a component D is about a query Q. Then its parent is $D \otimes D'$ with D' representing the siblings. With LMU, then also $D \otimes D' \Box\!\!\rightsquigarrow Q$, and exhaustivity is promoted to the ancestors. At the same time, we cannot increase specificity by adding information to the document component, as Right Monotonic Union does not hold, which we show next.

Regarding **Right Monotonic Union**, from $S \Box\!\!\rightsquigarrow T$, we cannot conclude $S \Box\!\!\rightsquigarrow T \otimes U$. Say, that $S \equiv map(A)$, $T \equiv map(B)$ and $T \otimes U \equiv map(C)$. From the premises and the definition of map, we then have $map(B) \subseteq map(C)$ and $map(A) \supseteq map(B)$. That does not necessarily mean that $map(A) \supseteq map(C)$. RMU is not supported.

Pure type XML retrieval is insensitive to the document length, but sensitive to the query length. The former is the case as we can extend the XML document by appending document components without changing the hierarchical inclusion of a query in a specific document component. If we change the query, we change the inclusion.

That left monotonicity holds, but right monotonicity does not, has interesting consequences for the behaviour of pure type XML retrieval with respect to the two XML evaluation dimensions of specificity and exhaustivity. Regarding exhaustivity, left monotonicity means that we can create more or at least as exhaustive answers by adding new document components. This behaviour reflects the characteristics of XML retrieval that parent elements are always as least as exhaustive answers as their children. We cannot create more exhaustive answers, however, by adding information to the query situations, as right monotonicity does not hold. By adding information to a query situation we create a more specific answer. We increase the focus. This is what Left Monotonic Union tells us if we take S and U to be query situations and T to be the document component situation. At the same time, we cannot increase specificity by adding information to the document component, as Right Monotonic Union does not hold.

The next combination rule **Cut** is also given. The assumption is in this case that $S \otimes T \Box\!\!\rightsquigarrow U$ and $S \Box\!\!\rightsquigarrow T$ are supported premises, while the conclusion is $S \Box\!\!\rightsquigarrow U$. We define $S \equiv map(A)$, $T \equiv map(B)$ and $U \equiv map(C)$. According to map, we interpret $\otimes$ as $\cup$. Then, $map(A) \cup map(B) \supseteq map(C)$ and $map(A) \supseteq map(B)$ lead to $map(A) \supseteq map(C)$. $S \Box\!\!\rightsquigarrow U$ is given and Cut holds. Both exhaustivity and specificity are equally influenced by this quality. For exhaustivity, we can say that if one document $D1$ is about another document $D2$ and both are about a query Q, the most exhaustive answer is $D1$. For specificity the same applies for a query $Q1$ that is about another query $Q2$. Then D is most specific to $Q1$.

Right Weakening does not hold, as the following example shows: Say S would be $\{ \langle\langle ElementType, Article, i_1; 1\rangle\rangle, \langle\langle Parent, i_1, i_2; 1\rangle\rangle, \langle\langle ElementType, Paragraph, i_2; 1\rangle\rangle, \langle\langle Value, garden, i_2; 1\rangle\rangle \}$, T would be $\{ \langle\langle ElementType, Paragraph, i_2; 1\rangle\rangle, \langle\langle Value, courtyard, i_2; 1\rangle\rangle \}$, and U would be $\{ \langle\langle ElementType, Article, i_1; 1\rangle\rangle, \langle\langle Parent, i_1, i_2; 1\rangle\rangle, \langle\langle ElementType, Paragraph, i_2; 1\rangle\rangle, \langle\langle Value, garden, i_2; 1\rangle\rangle \}$. Then, $S \Box\!\!\rightsquigarrow T \otimes U$, but not $S \Box\!\!\rightsquigarrow T$.

If **Mix** held, we would be able to conclude $S \otimes T \Box\!\!\rightsquigarrow U$, given $S \Box\!\!\rightsquigarrow U$ and $T \Box\!\!\rightsquigarrow U$. We define $S \equiv map(A)$, $T \equiv map(B)$ and $U \equiv map(C)$. Then, with $map(A) \supseteq map(C)$ and $map(B) \supseteq map(C)$, also $map(A) \cup map(B) \supseteq map(C)$. Thus, Mix is given, which is trivial, as Left Monotonic Union holds. For exhaustivity and specificity of interest is the fact that if two document components are both exhaustive answers to a query, together they form an exhaustive answer to the query. The same applies to specific document components. They stay specific to the combination of two queries.

Context-Free And is given as well. We can say that from $S \Box\!\!\rightsquigarrow T$ and $S \Box\!\!\rightsquigarrow U$ also $S \Box\!\!\rightsquigarrow T \otimes U$. Say, $S \equiv map(A)$, $T \equiv map(B)$ and $U \equiv map(C)$. As $map(A) \supseteq map(B)$ and $map(A) \supseteq map(C)$, also $map(A) \supseteq map(B) \cup map(C)$. Context-Free And is supported. It is supported, though Right Monotonic Union is not, as it controls the right monotonic behaviour by demanding aboutness for the added information. The document component must be about the expansion part of the query. This assumption ensures that the expanded query is still hierarchically included in the document component. Therefore, Context-Free And is supported by pure type XML retrieval. This means that we can increase specificity by adding new information to the document component if we know that this new information is also a specific answer in itself. Also, we can extend the query and might increase exhaustivity if we know that the query expansion is as well about the document component.

The **Containment** rule itself does not hold for pure type XML retrieval according to the definition of $S_i \rightarrow T_i$. Let us assume that $S_i \equiv \{ \langle\langle ElementType, Section, i_1; 1\rangle\rangle$, $\langle\langle Parent, i_1, i_2; 1\rangle\rangle$, $\langle\langle ElementType, Paragraph, i_2; 1\rangle\rangle$, $\langle\langle Value, garden, i_2; 1\rangle\rangle$, $\langle\langle Value, house, i_2; 1\rangle\rangle \}$ contains $T_i \equiv \{ \langle\langle ElementType, Paragraph, i_2; 1\rangle\rangle$, $\langle\langle Value, house, i_2; 1\rangle\rangle \}$. Say, that T_i is a subsituation of a situation T and $S_i \equiv S$. Then S is not about T. This confirms what we have said above about hierarchical inclusion as the basic aboutness relation for pure type XML retrieval. We cannot say that containment leads directly to aboutness, as it is just a necessary condition.

Absorption does not hold, as containment is defined over the element types. T might contain additional values compared to S, which are not absorbed under composition.

Right Containment Monotonicity does not hold. The assumptions that $S \Box\!\!\rightsquigarrow T$ and $T \rightarrow U$ do not allow for the conclusion of $S \Box\!\!\rightsquigarrow U$, because containment is only defined as structural containment. We could easily give an example so that U would not have a subset of the content of S though $T \rightarrow U$.

Non-conflict-containment is obviously given, because the definition of containment above implies the absence of preclusion. With respect to **Containment Preclusion**, if U precludes T, which is contained in S, then $S \perp U$. As pure type preclusion is defined over XML structure, S and U will together also not form a valid XML situation if $S \rightarrow T$. Containment Preclusion holds.

The first of the non-aboutness rules, **Mutual Preclusion**, is given, as document component and query are both XML documents. Either their structure means that they preclude each other, or there is no preclusion at all. **Simple Anti-Aboutness** allows to say from $S \Box\!\!\not\rightsquigarrow T$ that also $S \boxtimes\!\!\rightsquigarrow T$. This further statement can only be made using stronger semantics and is not covered by the XML structure. As said earlier, simple anti-aboutness is more an additional requirement than a logical property.

Negation Rational is given as we cannot add information to a query so that the information in the query is covered by the document component. E.g., a document component situation $D \, \{ \, \langle\langle ElementType, Article, i_1; 1\rangle\rangle \, , \, \langle\langle Parent, i_1, i_2; 1\rangle\rangle \, , \, \langle\langle ElementType, Para-graph, i_2; 1\rangle\rangle \, , \, \langle\langle Value, garden, i_2; 1\rangle\rangle \, , \, \langle\langle Value, house, i_2; 1\rangle\rangle \, \}$ is not about a query $\{ \, \langle\langle ElementType, Paragraph, i_2; 1\rangle\rangle \, , \, \langle\langle Value, flat, i_2; 1\rangle\rangle \, \}$. Then it will also never be about any extension of this query. If Negation Rational holds, so will **Strict Negation Rational**.

The **Closed World Assumption** allows to conclude $S \Box\!\!\not\rightsquigarrow T$, given $S_i \not\to_s T_j$. It holds for pure type XML retrieval, because two XML documents that are not contained in each other according to their XML structure cannot be about each other.

Guarded Left Monotonicity is interesting, as it is the first of what we called conservative monotonicity rules. Here, information composition is 'guarded' to avoid combining two precluding situations. As already stated semantic preclusion is beyond the scope of an investigation into XML retrieval in INEX. We can, however, look into a surface preclusion. Then, Guarded Left Monotonicity must hold, as left monotonicity holds as well. With the guard, however, we can avoid adding meaningless information to either increase exhaustivity by adding new information to the document side or increase the focus by using more relevant information for the query. **Guarded Right Monotonicity** will not hold, as Right Monotonic Union does not hold.

Qualified Left Monotonicity derives from the premises $S \Box\!\!\rightsquigarrow T$ and $T \not\perp U$ that $S \otimes U \Box\!\!\rightsquigarrow T$. As much as its sibling, Qualified Right Monotonicity, it is more interesting in more semantically oriented reasoning. In our case nothing can be said about the relationship of S and T if $T \not\perp U$ is known. Therefore, Qualified Left Monotonicity is just a special case of Left Monotonic Union and holds for pure type XML retrieval aboutness. Thus, **Qualified Right Monotonicity** is also a special case of Right Monotonic Union and is therefore not given for pure type XML retrieval aboutness.

Next, the completeness of the above rules needs to be proven.

4.7.5 Completeness

The completeness proof demonstrates that any possible aboutness situation is covered within our reasoning system. We have to demonstrate that for any aboutness relationship between two XML documents, their corresponding representation as situations leads to $\Box\!\!\rightsquigarrow$. For all valid XML documents A and $B \in \mathcal{X}$: If A about B then $map(A) \Box\!\!\rightsquigarrow map(B)$.

Proof Let $S \equiv map(A)$ and $T \equiv map(B)$. Also, let C be the subdocument of A if we remove B and $U \equiv map(C)$. $T \Box\!\!\rightsquigarrow T$, as Reflexivity is given. With Left Monotonic Union, $T \otimes U \Box\!\!\rightsquigarrow T$. With Set Equivalence and the definition of map, $S \Box\!\!\rightsquigarrow T$.

Having shown the completeness of our aboutness reasoning system, the reflection of pure type XML retrieval completes our discussion.

4.7.6 Reflection

For XML retrieval, we further specify the top and bottom document components and queries from Section 4.5 by differentiating those that are either exhaustive ($D \Box\!\!\leadsto Q$) or specific ($Q \Box\!\!\leadsto D$). We then have eight cases to cover, which we present in a simplified notation. For a complete overview of all eight cases see Table 5.9 in Section 5.2.6. Here, we only present four cases, which have an entry in the table for pure type XML retrieval. Let $\mathcal{D}$ be a set of document components and $\mathcal{Q}$ be a set of queries:

1. A top exhaustive document component D_j is always exhaustively about any query Q: $\{D_j | \forall Q, D_j \in \mathcal{D}, D_j \Box\!\!\leadsto Q\}$. The (virtual) root of the document collection is D_j, as exhaustivity is promoted up the document forest.

2. A top exhaustive query Q_j is the one which all document components D are exhaustive answers to: $\{Q_j | \forall D, Q_j \in \mathcal{Q}, D \Box\!\!\leadsto Q_j\}$. The top exhaustive query is $\{\emptyset\}$. Let us assume the document component D is itself $\{\emptyset\}$. Then, the only query Q that any D is always an exhaustive answer to is $\{\emptyset\}$. Let us assume $D \not\equiv \{\emptyset\}$. Then, with $D \equiv \mathit{map}(A)$ the only always given subset to $\mathit{map}(A)$ is $\{\emptyset\}$. Therefore, $\{\emptyset\}$ is the top exhaustive query.

3. A top specific document component D_j is always specifically about any query Q: $\{D_j | \forall Q, D_j \in \mathcal{D}, Q \Box\!\!\leadsto D_j\}$. The top specific document component is $\{\emptyset\}$. The proof is analogous to the one for top exhaustive queries.

4. A top specific query Q_j is the one which all document components D are specific answers to: $\{Q_j | \forall D, Q_j \in \mathcal{Q}, Q_j \Box\!\!\leadsto D\}$. The top specific query is again the (virtual) root of the document collection.

All the other entries in Table 5.9 are missing, as they are complementary statements. E.g., if a top exhaustive document component can be found, it is impossible that there is a bottom exhaustive query that will never find any answer in the document component set. We use this complementarity for our reflections of XML retrieval models to effectively reduce the number of reflections we have to do.

It might be surprising that the top specific query is like the top exhaustive document component the one that contains all the information in the document component set. This is the case as we are not looking for the most specific query — a question impossible to answer for all possible situations —, but we are looking for the one that delivers only specific results. This can just be the complete document tree, as all document components contain never more information than is present in the document tree.

4.8 Conclusion

This chapter has introduced our theoretical evaluation methodology. We have started with existing theoretical evaluation methodologies and have adjusted them to the requirements of XML retrieval. Our methodology is based on a well-defined number of steps, through which we iterate each time we analyse a model. The first step is the translation to define a model's symbolic representation of information as the result of its indexing mechanism. It is formally represented by the function *map*.

The next step in our methodology derives aboutness rules to describe the functional behaviour of XML retrieval systems. We have defined basic rules, combination and containment rules and also non-aboutness reasoning though the latter are seldom used in IR reasoning.

Aboutness is defined as a relationship between situations. In a theoretical evaluation framework, rules are used to define the reasoning aspects of this relationship. Rules are the logical representation of how a system decides a document to be about a query. Rules do not hold for all aboutness decisions but only for particular ones. Thus, an aboutness decision can be specified by the reasoning rules it incorporates. The aboutness decision can be further qualified by analysing how these reasoning rules are implemented by it: fully, conditionally or not at all. [Wong et al., 2001] call this functional benchmarking.

By comparing the kind of rules a particular system incorporates and the way it does so, we are able to give an overall comparison of the behaviour of XML retrieval systems. A further investigation of aboutness boundaries for particular retrieval systems is called reflection, our third step of each theoretical evaluation. Translation, reflection and aboutness rules were developed in [Huibers, 1996] as part of the theoretical evaluation of any retrieval model. In XML retrieval, we are also interested in how much a retrieval system uses structure to support the aboutness decision. Theoretically, we measure this by determining the difference in reasoning of the XML retrieval model to its 'flat' retrieval model equivalent (if there is one) and what we call pure type XML retrieval.The final step in our theoretical evaluation is the development of the pure type XML retrieval model to qualify the impact of structure on the retrieval performance.

The next chapter uses this methodology to investigate models from INEX.

Chapter 5

Theoretical Evaluation of XML Retrieval Models

5.1 Introduction

This chapter goes through five XML retrieval models submitted to INEX and evaluates them theoretically using the methodology developed in Chapter 4. We are looking only at models that performed well in INEX and are therefore comparable. Furthermore, all of the models performed not just during INEX 2005 but over a longer period of time so that one can assume that models are well developed and potential problems we find are not the result of a premature submission.

For each of the models, we first describe its background including its retrieval algorithm and indexing mechanism. Secondly, we calculate an example that reflects various standard retrieval situations. This allows us to understand better the overall behaviour of the model. Thirdly, we proceed with our theoretical evaluation of XML retrieval by first presenting the equivalent flat document retrieval model, before in the forth step, we iterate through all the theoretical evaluation steps described in Section 4.2: translation, aboutness rules, completeness and reflection. We repeat this procedure for each model, starting with the XML vector space retrieval model (Section 5.2). In Section 5.3, we analyse two language models and finally in Section 5.4 two structured models are introduced, which have been specifically designed for INEX.

5.2 XML Vector Space Retrieval

5.2.1 Background

As early as for INEX 2003, Mass and Mandelbrod present in *Retrieving the most relevant XML Components* an approach to XML retrieval that is based on the vector space model [Mass and Mandelbrod, 2005]. During INEX 2004 and 2005, they added new functionality to their original algorithm without changing its fundamental principles. Their idea is to use the vector space model for XML retrieval and rank document components instead of entire documents.

In order to adjust to the requirements of XML retrieval, [Mass and Mandelbrod, 2005] create a different index for each pre-defined component type. Six indexes are created according to the set of document components, most commonly seen as relevant in past INEX assessments. These are $\{article, abs, sec, ss1, ss2, p, ip1\}$. The article contains the complete XML document, sec all section elements and so on.

One advantage of this approach is its layered approach. It can be built on top of almost any existing IR model, as it takes each document component to be a document in itself. At runtime, queries can work on each index in parallel. One challenge is that the indexing here destroys the unique position of an XML element. Therefore, we cannot determine afterwards, which section element was the parent of which paragraph element, etc. From the index structure there is no way back to rebuild the original XML structure. The authors address this by storing all the structural information within the article index and then do some post-processing on top of the initial retrieval results.

From INEX 2004 onwards [Mass and Mandelbrod, 2005], the retrieval status value is defined by, were D and Q are document components and query:

$$rsv(D,Q) = \frac{\sum_{t_i \in \{Q \cap D\}} w_Q(t_i) * w_D(t_i) * idf(t_i)}{\|Q\| * \|D\|} \tag{5.2.1}$$

$$idf(t) = log(\frac{|D|}{|D(t)|})$$

$$w_Q(t) = \frac{log(TF_Q(t))}{log(AvgTF_Q)}$$

$$w_D(t) = \frac{log(TF_D(t))}{log(AvgTF_D)}$$

$$idf(t) = log(\frac{NumberOfDocumentsInCollection}{DocumentsContaining(t)})$$

$TF_Q(t)$ stands for the number of occurrences of a term t in Q, while $TF_D(t)$ describes the number of occurrences of t in D. $AvgTF_Q$ captures the average number of occurrences of all query terms in Q and $AvgTF_D$ does the same for D. $||Q||$ is the number of unique terms in Q and $||D||$ is the number of unique terms in D. Both are scaled by the average document length in the collection.

For each index a query produces a list sorted by the relevance of the elements [Mass and Mandelbrod, 2005]. The scores of the indexes are index-independent normalized into the range [0;1]. This overall normalisation is achieved by a division with $rsv(Q,Q)$, by calculating the retrieval status value of the query as if it would have been part of the document collection. Each index entry is normalised towards $rsv(Q,Q)$ as the maximum value. The sorted and normalised index entries are afterwards merged into one list that combines all granularities.

A priori dividing the different XML elements into separate indexes, solves the problem of nested components, but it lacks for each component index context information, as each component is treated *as if* it would be a document in itself. This simplifies the XML retrieval problem, but leads to a number of issues that have to be addressed by additional processing. The authors found, e.g., that they had a problem due to malformed index statistics [Mass and Mandelbrod, 2005]. The fine grained indices do not deliver information outside their scope. For example, the articles index contains 42,578,569 tokens while the paragraphs index contains only 31,988,622 tokens [Mass and Mandelbrod, 2005]. For paragraphs, $\tilde{2}5\%$ of statistics is missing. Yet, a term with a low document frequency (df) based on the indexed tokens may actually be quite frequent outside the paragraphs so that its df should be higher. In 2004, Moss and Mandelbrod add a variation to their original system, that provides a solution for this problem using a document pivot [Mass and Mandelbrod, 2005]. They normalise each score by the containing article score with the following formula where S_a is the containing article score and S_c the component score: $DocPivot * S_a + (1 - DocPivot) * S_c$. $DocPivot$ is an additional constant.

On top of $DocPivot$, Moss and Mandelbrod apply Automatic Query Refinement (AQR) to the component ranking algorithm. In their AQR, the query is run in two rounds, where highly ranked results from the first round are used to add new query terms and to reweigh the original query terms in the second round. Now, the rsv is calculated as follows:

1. For each index i
   (a) Compute the result set Res_i of running a query Q on index i.
   (b) Apply Automatic Query Refinement on Res_i.
   (c) Normalize scores in Res_i by applying $rsv(Q, Q)$.
   (d) DocPivot: Scale each rsv by its containing article rsv from Res_0.
2. Merge all Res_i to a single result set Res composed of all components sorted by their score.

The Automatic Query Refinement, the authors apply, follows the idea of Lexical Affinity (LA) terms. These amend the existing query to achieve better results. They are chosen according to the degree they separate relevant from non-relevant document components. Lexical Affinity describes pairs of terms where exactly one of the pair is part of the query and both appear close in relevant documents. Four parameters (M, N, K, α) determine the overall ranking, with M denoting the number of highly ranked documents to use for constructing a list of candidate LA's while N $(N >> M)$ describes the number of highly ranked components to be used for selecting the best K LA's (among the candidate LA's). These are those which have the highest information gain (IG) (depending on the further tuning parameter α).

$$IG_{D,Q}(L) = H_Q(D) - [\frac{|D^+|}{|D|}H_Q(D^+) + \frac{|D^-|}{|D|}H_Q(D^-)] \tag{5.2.2}$$

$$H_Q(x) = -p_Q(x)log(p_Q(x)) - (1 - p_Q(x))log(1 - p_Q(x)) \tag{5.2.3}$$

The IG determines how much a lexical affinity L is able to discriminate relevant from non-relevant documents. It is calculated using parameters D^+ and D^- denoting the set of document components having the lexical affinity L or respectively not having it. $H_Q(x)$ describes the level of disorder (entropy) of a document component. We are interested in those elements that optimize the entropy of relevant documents. $H_Q(D)$ is a constant, as it is independent of L, and can therefore be omitted. $H_Q(D^+)$ stands for the entropy of relevant documents, which is determined by using $p_Q(x) = \frac{|R^+|}{|D^+|}$, where R^+ are the relevant document components in D^+. As we do not know R^+, we need to estimate it. The scoring function is used as an approximation. $H_Q(D^-)$ can be found analogously. The authors take $p_Q(D^+)$ to be sum of scores of documents in D^+ normalised by $|D^+|$, and similarly for $p_Q(D^-)$. This way the LA's with the highest information gain are found and then added to the query Q. The scores for the query are recalculated with $w_Q(t) * w_D(t) * idf(t)$.

Next, we calculate an example to see in more detail the impact of the different steps of the ranking function on the overall results. We use the same example for all our theoretical

Table 5.1: Number of unique tokens

Q	D1	D2	D3	P_{11}	P_{12}	P_{21}	P_{22}	P_{31}	P_{32}
3	4	5	4	2	2	2	3	1	3

evaluations, which ensures better comparability. The example has been chosen carefully to reflect several standard retrieval situations such as a small document component that is only about the query but does not cover all of it or a large document component that is exhaustively about the query but has other additional information.

5.2.2 Example

In the example, let the query situation be about 'house', 'garden' and 'courtyard'. Let document situation $D1$ be the parent of the paragraphs P_{11} and P_{12}. Furthermore, $D2$ is the parent of P_{21} and P_{22} and $D3$ is the parent of P_{31} and P_{32}. P_{11} is about 'garage' and 'house', P_{12} about 'damage' and 'fire', and $D1$ about the combination of both. Let P_{21} be about 'door' and 'garden' and P_{22} be about 'arrive', 'garden' and 'courtyard', with $D2$ again being the combination of the two. Finally, P_{31} is about 'house' and P_{32} is about 'garage', 'arrive' and 'courtyard'. This completes our example.

Table 5.2: $W_D(t)$

	arrive	damage	door	fire	house	garage	garden	courtyard
w_Q	0	0	0	0	1	0	1	1
w_{D1}	0	1	0	1	1	1	0	0
w_{D2}	0.77	0	0.77	0	0	0	1.73	0.77
w_{D3}	1	0	0	0	1	1	0	1
$w_{P_{12}}$	0	0	0	0	1	1	0	0
$w_{P_{22}}$	0	1	0	1	0	0	0	0
$w_{P_{21}}$	0	0	1	0	0	0	1	0
$w_{P_{22}}$	1	0	0	0	0	0	1	1
$w_{P_{31}}$	0	0	0	0	1	0	0	0
$w_{P_{32}}$	1	0	0	0	0	1	0	1

Table 5.3: IDF(t)

	arrive	damage	door	fire	house	garage	garden	courtyard
Ind_1	0.18	0.48	0.48	0.48	0.18	0.18	0.48	0.48
Ind_2	0.48	0.78	0.78	0.78	0.48	0.48	0.48	0.78

Table 5.1 shows the number of unique tokens per document component. For w_Q in (5.2.1), we add the constant 0.5 to both parts of the fraction to avoid $log(1) = 0$, which is common for our small example, where the term frequency of 1 is relevant. Therefore, we calculate:

$$w_Q(t) = \frac{log(TF_Q(t) + 0.5)}{log(AvgTF_Q) + 0.5)}$$

$$w_D(t) = \frac{log(TF_D(t) + 0.5)}{log(AvgTF_D + 0.5)}$$

$$idf(t) = log(\frac{NumberOfDocumentsInCollection}{DocumentsContaining(t)})$$

Table 5.4: rsv(D,Q)

	$D1$	$D2$	$D3$	P_{11}	P_{12}	P_{21}	P_{22}	P_{31}	P_{32}
$rsv(Q, D)$	0.17	0.64	0.59	0.64	0.0	0.64	1.18	4.03	1.09

These calculations deliver the results shown in Tables 5.2 and 5.3.

Table 5.5: $IG(D, Q)$

(house,arrive)	(garden,arrive)	(courtyard,arrive)	(garden, door)	(house, garage)	(courtyard, garage)
-0.098	-0.096	-0.086	-0.096	-0.096	-0.098

Table 5.6: score(D,Q')

	$D1$	$D2$	$D3$	P_{11}	P_{12}	P_{21}	P_{22}	P_{31}	P_{32}
$rsv(D, Q')$	0.08	0.67	0.42	0.50	0.0	0.50	1.18	1.01	0.92

Table 5.4 gives $rsv(D, Q)$, where D is a document component and Q is the query.

As in [Mass and Mandelbrod, 2005], we assume $M = 2$. Then candidate LA's come from $D3$ and $D2$ and are part of the set: $\{(house, arrive); (garden, arrive); (courtyard, arrive); (house, door); (garden, door); (courtyard, door); (house, garage); (garden, garage); (courtyard, garage)\}$. [Mass and Mandelbrod, 2005] furthermore assumes that $N = 2$ and thus rsv has to be $n > 0.02$. This leads us to the information gains IG, as in Table 5.5.

With $K = 1$ and $\alpha = 0.9$ according to [Mass and Mandelbrod, 2005], we add 'arrive' to the query. The enhanced query situation Q' is $\{\, \langle\langle house \rangle\rangle\,, \langle\langle garden \rangle\rangle\,, \langle\langle courtyard \rangle\rangle\,, \langle\langle arrive \rangle\rangle\,\}$. This is a better choice, as it clearly excludes $D1$. Yet, there is a problem for more specific answers $Q \,\Box\!\!\leadsto D$. The new LA can never be part of the original query (according to the assumptions). Therefore, the added token cannot improve $Q \,\Box\!\!\leadsto D$ or the focus. E.g.: If we add the query term 'garden' to a query, then this query will be less specific about a component having information about 'house' and 'car'. This is intrinsic to the model according to the assumptions of AQR.

Table 5.6 shows the new scores for $rsv(D, Q')$. P_{32} is now regarded more relevant. This has to be the case, as it contains 'arrive'. Even more importantly, P_{22} now tops the list. All its items are relevant and it contains more information than the former most relevant element P_{31}. Finally, we normalise the results by calculating $rsv(Q', Q')$. Table 5.7 shows the results.

The normalisation has clearly changed the order. The highly specific and exhaustive document component P_{22} is still the highest ranked element. But $D2$ has now overtaken P_{31} and P_{32} and is closer to P_{22}. All of these changes are indications of the major importance of normalisation in XML retrieval, as a way to compare components of different information size.

The last step is $DocPivot$. We use $DocPivot = 0.5$, as the authors used for their 2005 experiments [Mass and Mandelbrod, 2005]. Table 5.8 summarizes the results.

Table 5.7: Normalized rsv(D,Q')

	$D1$	$D2$	$D3$	P_{11}	P_{12}	P_{21}	P_{22}	P_{31}	P_{32}
$rsv(D,Q')$	0.09	0.73	0.46	0.32	0.0	0.32	0.76	0.65	0.61

Table 5.8: Final rsv(D,Q)

	$D1$	$D2$	$D3$	P_{11}	P_{12}	P_{21}	P_{22}	P_{31}	P_{32}
$rsv(D,Q)$	0.09	0.73	0.46	0.21	0.05	0.53	0.75	0.56	0.54

Obviously, with $DocPivot = 0.5$ the score for the containing articles does not change. In the overall ranking, P_{22} has fostered its position. It has now clearly left P_{31} and P_{32} behind, as its containing article $D2$ is the most relevant one. P_{31} and P_{32} are still more relevant than their containing article $D3$. The overall ranking is now:

$$P_{22}, D2, P_{31}, P_{32}, P_{21}, D3, P_{11}, D1, P_{12}$$

These are very good ranking results considering the original query (especially in the top two places). But also, the preference for the more specific P_{31} and P_{32} rather than $D3$ is convincing. $D1$ and any of its subelements do not play a role at all. The scores have clearly improved. We can also see, however, that with AQR more specific answers are ranked lower than more exhaustive answers, as the newly added information is never part of the original topic items. The most specific answer P_{31} has lost two places compared to the original score.

The next step in our theoretical evaluation is the translation.

5.2.3 Translation

This section defines the translation given a set of XML elements $\mathcal{D}$ to a situation S. It is based on the translation of the flat document vector space model, as defined in Section 4.6.2, because document components are taken as if they were full documents. Only the most informative elements are indexed and all results are merged into a final result list.

Let d be an XML element in $\mathcal{D}$. $\chi(d)$ is a set of descriptors, which contain the index terms as well as all other information necessary to calculate rsv such as the term frequency. These descriptors correspond to the non-zero entries in the vectors for d, while $\chi(q)$ corresponds to non-zero entries for a query q. Then, map describes all element situations that are most informative:

$map(\chi(d)) = \{ \langle\langle ElementType, e, i; 1\rangle\rangle, \langle\langle Value, t, i; 1\rangle\rangle | e \in \{article, abs, sec, ss1, ss2, , p, p1\}, t \in \chi(d))\}$

e is an element type, t a term and i and identifier parameter. Terms can be found in content infons of these element types, while each $\chi(d)$ describes document components that are part of the most informative XML elements. That $\chi(d)$ describes components instead of whole documents, marks the decisive difference from the traditional vector space model. The translation of a query to a query situation is defined in a similar way.

Next we need to define operators used in the rules from Section 4.4: equivalence, composition, containment and preclusion. As our translation qualifies the simple vector space one using element types and descriptors, we can reuse the ones for flat document vector space retrieval from Section 4.6.2. Then, given two situations S and T, they are equivalent if all their infons are identical, and they can be composed using $\cup$ and parameter replacement. S contains T, if T has only infons from S. We thus define containment as surface containment. Deep containment is an addition to this vector space model. Only preclusion differs slightly from what we have seen in 4.4. Now, a situation S describing a document component also precludes another situation T if one of them has an element type that is not part of the most informative elements. This is a structural kind of preclusion, similar to what we have seen in Section 4.7 for pure type XML retrieval.

5.2.4 Rules

As each XML element is inserted separately in the index, the main difference to flat vector space retrieval is that we consider elements instead of full documents. Given a document d and a query q (indexed by descriptor sets $\chi(d)$ and $\chi(q)$), the *XML retrieval vector space aboutness decision* is defined by:

$$d \text{ about } q \text{ if and only if } rsv(\chi(d), \chi(q)) \geq n$$

For the AQR step only the top N documents are considered. We call the value that has to be reached in order to be part of the top N documents n. Thus, the model implements thresholded vector space retrieval [Wong et al., 2001], as described in Section 4.6.

We now continue with analysing the functional properties of the model. We cannot find a proposition that would relate rsv to a set-theoretical equation, as we could do, for instance, for the pure type model in Proposition 4.7.1. We therefore need to argue directly with the aboutness decision based on rsv and look at its mathematical composition. We need to discuss how the individual components of the rsv equation influence the overall calculation when they are changed according to the assumptions of our reasoning rules. This is a proven method, common to many social science analyses [Brunn, 2007].

For vector space retrieval in general, we would like to exclude Reflexivity in order to avoid logical anomalities [Huibers, 1996]. **Singleton Reflexivity** is supported for XML vector space retrieval. In fact, it will be the maximum retrieval status value $rsv(\chi(q), \chi(q))$ and is used for the normalisation, as seen in Section 5.2.1. Singleton Reflexivity is fully supported.

The model is also **symmetric**. $rsv(\chi(d), \chi(q)) > n$ is equivalent to $rsv(\chi(q), \chi(d)) > n$, as in Equation (5.2.1) the numerator of the retrieval status value fraction is the sum of products depending on q and d and the denominator is a purely symmetrical product. Document component and query both have the same influence on the retrieval status value, as both are equal parts of the sum.

Set Equivalence is also given. We only show it for Left Set Equivalence. Then, we can conclude $T \Box\!\!\rightsquigarrow U$ from the assumptions $S \Box\!\!\rightsquigarrow U$ and $S \equiv T$. Let us assume that $S \equiv map(A)$, $T \equiv map(B)$ and $U \equiv map(C)$. A, B and C are sets of descriptors. Then, we know that $rsv(A, C) > n$. With $B \equiv C$ (they are the same descriptor set), we can state without contradiction that $rsv(B, C) > n$. Thus, $T \Box\rightsquigarrow U$ and Set Equivalence is supported.

XML vector space retrieval, however, does not support **Transitivity**. We can easily construct an example so that $rsv(\chi(d), \chi(q)) \geq n$ and $rsv(\chi(d'), \chi(q)) \geq n$ but not $rsv(\chi(d), \chi(d')) \geq n$. The terms t_i in Equation (5.2.1) responsible for $rsv(\chi(d), \chi(q)) \geq n$ and the terms t_j responsible for $rsv(\chi(d'), \chi(q)) \geq n$ do not have to be overlapping so that $rsv(\chi(d), \chi(d')) \geq n$.

As the aboutness decision is based on overlap of terms t_i and t_j, **Euclid** is not given either. With Euclid, from $S \Box\rightsquigarrow T$ and $S \Box\rightsquigarrow U$ we would be able to derive $T \Box\rightsquigarrow U$. Let us assume that $S \equiv map(A)$, $T \equiv map(B)$ and $U \equiv map(C)$. For the counterexample, let us assume that $n = 0$. Then obviously, $rsv(A, B) > 0$ as well as $rsv(A, C) > 0$. However, this does not necessarily include $rsv(B, C) > 0$, as U and T might not share terms.

Next, we discuss **Left Monotonic Union** (LMU). Does for any situations S and T, $S \Box\rightsquigarrow T$ imply that also $S \otimes U \Box\rightsquigarrow T$? LMU is conditionally given. We split the discussion of the involved calculations into two steps, the first step being the vector space based relevance calculation rsv and the second being the AQR step.

For the first step, let us assume that $S \equiv map(A)$, $T \equiv map(B)$ and $U \equiv map(C)$. Then, $S \Box\rightsquigarrow T$ means that $rsv(A, B) \geq n$. We are not interested in the details of the top part of Equation (5.2.1) and rewrite $rsv(A, B) = \frac{f(AB)}{||A|| * ||B||}$. $||.||$ stands for the number of unique tokens, while $f(AB)$ describes a function dependent on the informational overlap of A and B. We can then progress by analysing how the individual components of the Equation (5.2.1) relate to each other.

The behaviour of f depends on the following factors:

- $f(AB)$ will always be larger if there is more information overlap in A and B.
- $f(AB)$ will be larger if the frequencies for A and B are much higher than the averages, accordingly.
- $f(AB)$ will be larger if the elements in AB are not spread out across too many documents.

Then, the overall behaviour of $rsv(A, B)$ is determined by the size of $f(AB)$ in relation to the number of unique terms in A and B. It is clear that the more unique terms a document component has and the less it has a significant overlap with the query, the more difficult it is to pass the threshold n. This is appropriate for XML retrieval, as focussed document components have less unique terms. Looking at LMU, the newly added information on the left side of the aboutness relation can, however, have many unique terms. Then, amending $f(AB)$ with new terms could have a negative effect and the threshold would be missed.

Let us assume that Q_{AQR} is the added query part. In the example above it was $\{ \langle\langle arrive \rangle\rangle \}$. This is the query subsituation that is added to refine the initial query. The refined scores then deliver a larger overlap for those document components that were relevant in the first step, but this emphasis on relevant components only has a positive effect as long as the newly added information does not contribute to a significant increase of $||Q||$. The latter can easily happen, as only information, not present in the original query, is to be added in AQR. So, $||Q||$ has to increase. This increase can be outweighed, however, if the new information overlap is large enough. Being dependent on the number of unique terms in Q and D, LMU is only conditionally supported.

LMU is conditionally supported, while for pure type XML retrieval and the simple vector space model it was fully. This ensures a more conservative approach to monotonicity which means that aboutness is only preserved under certain conditions [Wong et al., 2001]. This allows for a better control of this important quality and adds to the model's convincing performance in the INEX campaigns [Gövert et al., 2006], which we will discuss in more detail in Chapter 8. However, the condition is chosen a priori by setting N, and is external to the aboutness decision, as we will see when we discuss the further reasoning behaviour.

Because of the symmetric composition of the rsv function **Right Monotonic Union** is just the mirror case of Left Monotonic Union. We say that $S \equiv map(A)$, $T \equiv map(B)$ and $U \equiv map(C)$. Again, $rsv(A, B) \geq n$, means that Right Monotonic Union is only conditionally supported. Both monotonic unions are therefore only conditionally supported.

Cut's conclusion is $S \,\Box\!\!\rightsquigarrow\, U$, given that $S \otimes T \,\Box\!\!\rightsquigarrow\, U$ and $S \,\Box\!\!\rightsquigarrow\, T$. We say that $S \equiv map(A)$, $T \equiv map(B)$ and $U \equiv map(C)$. According to the assumptions, $rsv(AC, B) \geq n$ and $rsv(A, B) \geq n$, where AC stands for the combination of A and C. This does not necessarily lead to $rsv(A, C) \geq n$, as the example of $map(A) \equiv \{\, \langle\langle ElementType, Paragraph, p; 1\rangle\rangle\,,\ \langle\langle Value, house, p; 1\rangle\rangle \,\}$, $map(B) \equiv \{\, \langle\langle ElementType, Paragraph, p; 1\rangle\rangle\,,\ \langle\langle Value, courtyard, p; 1\rangle\rangle\,,\ \langle\langle Value, house, p; 1\rangle\rangle \,\}$ and $map(C) \equiv \{\, \langle\langle ElementType, Paragraph, p; 1\rangle\rangle\,,\ \langle\langle Value, courtyard, p; 1\rangle\rangle \,\}$ demonstrates.[1] Cut is not supported.

Right Weakening is also not supported. Otherwise, we would be able to conclude $S \,\Box\!\!\rightsquigarrow\, T$ given that $S \,\Box\!\!\rightsquigarrow\, T \otimes U$. From $\{\, \langle\langle ElementType, p; 1\rangle\rangle\,,\ \langle\langle Value, garden, p; 1\rangle\rangle \,\}$ about $\{\, \langle\langle ElementType, p; 1\rangle\rangle\,,\ \langle\langle Value, house, p; 1\rangle\rangle\,,\ \langle\langle Value, garden, p; 1\rangle\rangle \,\}$, we cannot conclude $\{\, \langle\langle ElementType, p; 1\rangle\rangle\,,\ \langle\langle Value, garden, p; 1\rangle\rangle \,\}$ about $\{\, \langle\langle ElementType, p; 1\rangle\rangle\,,\ \langle\langle Value, house, p; 1\rangle\rangle \,\}$. Right Weakening is not supported.

Mix was presented in Section 4.4.2 as a specific variant of Left Monotonic Union and is therefore at least conditionally supported. It makes sure that the added information is about the target situation. Investigating Left Monotonic Union, we have argued above that a threshold n must be reached, though the added information might add too many unique terms in query and document component. Let us assume that $S \equiv map(A)$, $T \equiv map(B)$ and $U \equiv map(C)$. Mix implies that $rsv(A, C) \geq n$ as well as $rsv(B, C) \geq n$. In Equation (5.2.1) only new terms are added that are part of $Q \cap D$. Under no circumstance can the sum in the denominator decrease. Yet, this does not change that $||Q||$ and $||D||$ might increase so that then rsv will fall below n. This demonstrates that Mix is only conditionally supported as well as that the control of the monotonic behaviour for the XML vector space retrieval model is not dependent on the actual information overlap, as

[1] To enhance readability, we ignore all additional information next to the index terms in the descriptors such as term frequency, etc.

N is chosen a priori.

While Mix is a variant of Left Monotonic Union, **Context-Free And** is a specific variant of Right Monotonic Union. What has been said about Mix, also applies to Context-Free And. It is conditionally supported, as the change in $||Q||$ and $||D||$ might outweigh the information overlap increases.

Next, the containment rules will be looked at. Deep containment is not given for the XML vector space model. The query refinement can be seen as introducing some kind of semantics, but it remains on the level of co-occurrences of terms. **Containment** does not hold for the XML retrieval vector space model. Let us assume that $S \equiv map(A)$ and $T \equiv map(B)$, as well as $S_i \equiv map(C)$ and $T_j \equiv map(D)$. S_i is about T_j only if the size of T_j is large enough to constitute an overlap that makes $rsv(C, D) \geq n$. As S_i is a subsituation of S and T_j of T, also $rsv(A, B) \geq n$ only if $rsv(C, D) \geq n$. Containment is therefore not necessarily given for the model.

Absorption holds, too, for similar reasons as for the plain vector space model. As containment means that all the infons in S can also be found in T, composing S and T means absorbing the infons in T, as composition uses $\cup$. Absorption is therefore supported.

Right Containment Monotonicity is a variant of Right Monotonic Union. It concludes that $S \Box\!\!\rightsquigarrow U$, given that $S \Box\!\!\rightsquigarrow T$ and $T \rightarrow U$. Say, $S \equiv map(A)$, $T \equiv map(B)$ and $U \equiv map(C)$. Then, it is not necessarily the case that $S \Box\!\!\rightsquigarrow U$. Say $rsv(A, C) = ac$ and $rsv(A, B) = ab$. According to the assumption of containment $ac \leq ab$. Thus, though $ab \geq n$, not necessarily also $ac \geq n$. Right Containment Monotonicity is not supported.

Non-conflict-containment is trivially given, because we assume preclusion to be a relation only between elements of different types where one is not informative. If two subsituations contain each other, they cannot have different types and do not preclude each other. Similarly, **Containment Preclusion** is given. If a situation S contains T, then they must have the same element type. If T precludes U, then they must have different element types where one is not informative. Thus, S must also preclude U, and Containment Preclusion is supported.

The first of the non-aboutness reasonings is **Mutual Preclusion**, which is trivially given if we define preclusion only structurally. If S precludes T, one of them must represent elements that are not informative. Thus, T also precludes S. **Simple Anti-Aboutness**, however, is not supported. Just because two situations are not about each other, it does not mean that their element types are not informative.

Negation Rational is not strict enough for the XML vector space retrieval model. We need to have an overlap of significantly relevant information items. According to the assumption of Negation Rational: With $S \equiv map(A)$, $T \equiv map(B)$ and $U \equiv map(C)$, we can without contradiction say that $rsv(A, B) = 1$ and therefore $S \Box\!\!\not\rightsquigarrow T$, while $rsv(AC, B) \geq 2$ and therefore $S \Box\!\!\not\rightsquigarrow T \otimes U$. 1 could be below the threshold but 2 above. Thus, Negation Rational is not given. **Strict Negation Rational** cannot hold, as Negation Rational does not.

The **Closed World Assumption** improves precision. It clearly does not hold for the XML retrieval vector space model. Though two XML elements do not contain each other, they still might overlap enough in their information to pass the threshold in rsv.

The last set of rules that we are going to investigate are the conservative aboutness rules. They are all trivial extensions of the unguarded monotonicity rules. **Guarded Left Monotonicity** (GLM) will conclude $S \otimes U \Box\!\!\rightsquigarrow T$ if $S \Box\!\!\rightsquigarrow T$ and S does not preclude T. We assumed that S and T would only preclude each other, if they are situations of different element types. Without this assumption, we would have no monotonicity rule at all. Thus, GLM is just a specification of the reasoning assumptions already given for left monotonicity and holds conditionally. The same applies for **Guarded Right Monotonicity**. It holds conditionally, as it is a special case of Right Monotonic Union that makes the assumptions in it explicit.

Qualified Left Monotonicity and **Qualified Right Monotonicity** both hold conditionally as special cases of Left and Right Monotonic Union.

5.2.5 Completeness

Next, the completeness of the above rules needs to be proven. We have to show that if $rsv(A, B) \geq n$ then $map(A) \Box\!\!\rightsquigarrow map(B)$.

First, let us assume that in order to achieve $rsv(A, B) \geq n$, the overlap in items in query and documents in Equation (5.2.1) must be at least of counting size t. Let us further assume that $A \cap B \equiv C$. Obviously, $C \subseteq A$ and $C \subseteq B$, and $|C| \geq t$. Let $D \equiv A \setminus C$ and $E \equiv B \setminus C$. Furthermore, let us assume that $S \equiv map(A)$, $T \equiv map(B)$, $U \equiv map(C)$, $V \equiv map(D)$ and $W \equiv map(E)$. Then also with Singleton Reflexivity and LMU:

$$\frac{U \Box\!\!\rightsquigarrow U}{U \otimes W \Box\!\!\rightsquigarrow U}$$

We can apply LMU without further condition, as we know from the assumptions that $|C| \geq t$. Then, also $|C \cup E| \geq t$. Next, we apply Set Equivalence:

$$\frac{U \otimes W \Box\!\!\rightsquigarrow U, T \equiv U \otimes W}{T \Box\!\!\rightsquigarrow U}$$

Using, Symmetry and LMU again, we can derive:

$$\frac{T \Box\!\!\rightsquigarrow U}{U \Box\!\!\rightsquigarrow T}$$

$$\frac{U \Box\!\!\rightsquigarrow T}{U \otimes V \Box\!\!\rightsquigarrow T}$$

Again, LMU must again unconditionally hold. To arrive at $S \Box\!\!\rightsquigarrow T$, we can use Set Equivalence again:

$$\frac{U \otimes V \Box\!\!\rightsquigarrow T, S \equiv U \otimes V}{S \Box\!\!\rightsquigarrow T}$$

The aboutness proof system is complete, as we have a rule set to conclude that $S \Box\!\!\rightsquigarrow T$, given $rsv(A, B) \geq n$.

This proof is based on the coordinate retrieval one in [Huibers, 1996]. Yet, it is also different at the same time, which clearly shows the difference in his aims and objectives towards ours. As discussed, Huibers is interested in showing the soundness and completeness of principal retrieval models. We are interested in looking at actual retrieval models from INEX. His completeness proofs do not include the conditional aboutness rules, as these are unknown to his set of aboutness rules. In order to prove our completeness, we need to show that the conditions in the rules we are applying are fulfilled in our particular circumstances.

5.2.6 Reflection

For the reflection of XML vector space retrieval, we look at the eight cases developed in Section 4.7.6. Our aim is to compare the behaviour of this model with the pure type model based on hierarchical inclusion. The reflection gives us another measure of how much hierarchial inclusion is realised in the model.

The first case we consider is (1) the bottom exhaustive query. Let us assume, the query Q is itself $\{\emptyset\}$. Then, the only document component D that is never an exhaustive answer to Q is $\{\emptyset\}$. Let us furthermore assume, $Q \not\equiv \{\emptyset\}$. In this case, we cannot guarantee that no document component will ever be retrieved, as with Singleton Reflexivity Q will be at least about itself. Therefore, there is no other situation than $\{\emptyset\}$ that is part of the bottom exhaustive query set. Thus, the bottom exhaustive query is $\{\emptyset\}$. In rsv, then $||Q|| = 0$ and rsv becomes undefined. In an analogous manner, we can prove that (2) the bottom specific document component, (3) the bottom exhaustive document component and (4) the bottom specific query are all $\{\emptyset\}$. The results are summarized in Table 5.9.

Table 5.9: Reflection of structural behaviour of XML vector space model

	XML vector space Retrieval	Pure type XML retrieval
Top Exhaustive Document Component		(virtual) root
Top Exhaustive Query		$\{\emptyset\}$
Bottom Exhaustive Document Component	$\{\emptyset\}$	
Bottom Exhaustive Query	$\{\emptyset\}$	
Top Specific Document Component		$\{\emptyset\}$
Top Specific Query		(virtual) root
Bottom Specific Document Component	$\{\emptyset\}$	
Bottom Specific Query	$\{\emptyset\}$	

Looking at the pure type XML retrieval and XML vector space model in Table 5.9, where we have entries for the pure type XML retrieval, we miss them for the XML vector space model and vice versa. The result of the reflection is therefore that XML vector space retrieval does not include XML structure in its aboutness decision. This becomes particularly clear if we compare the XML vector space retrieval reflection with the one for pure type XML retrieval: The XML vector space retrieval model is first of all not able to discriminate the behaviour for the cases where we find bottom exhaustive and specific document components and queries. Everywhere the entry is $\{\emptyset\}$. In particular, the model does not deliver a concept of top specific document components, which would be a theoretical version of a document component that is always a focussed answer. This can be seen as a disadvantage if the declared aim of XML retrieval in general is to deliver the most specific answers. XML vector space retrieval does not deliver an approximation of what is always top specific and therefore does not have an element describing the boundaries of specificity.

The vector space model in itself has no means to express structure. The vectors do not contain dimensions that reflect the structural composition of the information they represent. That the flat model lacks an inherent means to express structure, entails that also the XML vector space retrieval model has no intrinsic way of expressing structure. This has become particularly clear while comparing the XML vector space retrieval reflection with pure type XML retrieval one.

The overall conclusion from Section 5.2.4 and this section is that the XML vector space retrieval model does not substantially differ from the flat based one. We will see in Chapter 8 how this fact manifests itself in the experimental evaluation.

5.2.7 Conclusion

Table 5.10 summarises the results of our theoretical evaluation for the vector space models. We can clearly see that XML vector space retrieval differs strongly from pure type XML retrieval. XML vector space retrieval is essentially based on information overlap between D and Q, not on their structural relationship.

Reasoning behaviour	*Plain vector space*	*XML vector space*	*Pure Type XML Retrieval*
Singleton Reflexivity	fully	fully	N/A
Reflexivity	N/A	N/A	fully
Symmetry	fully	fully	not
Set Equivalence	fully	fully	fully
Transitivity	not	not	fully
Euclid	not	not	not
LMU	fully	$\|\|D\|\|$ and $\|\|Q\|\|$	fully
RMU	fully	$\|\|D\|\|$ and $\|\|Q\|\|$	fully
Cut	not	not	fully
Right Weakening	not	not	not
Mix	fully	$\|\|D\|\|$ and $\|\|Q\|\|$	fully
Context-Free And	fully	$\|\|D\|\|$ and $\|\|Q\|\|$	fully
Containment	fully	not	not
Absorption	fully	fully	not
Right Containment Monotonicity	fully	not	not
Non-Conflict-Containment	N/A	fully	fully
Containment Preclusion	N/A	fully	fully
Mutual Preclusion	N/A	fully	fully
Negation Rational	N/A	not	fully
Closed World Assumption	N/A	not	fully

Table 5.10: Vector space retrieval evaluation results

If we compare the reasoning behaviour of XML vector space retrieval with its flat equivalent on the one hand side and with the pure type model on the other hand side, various differences in Table 5.10 are significant. XML vector space retrieval is a symmetric model following its flat equivalent, while pure type retrieval is asymmetric and supports transitive reasoning. These two key reasoning properties that indicate advanced reasoning using the XML structure are both not supported by the analysed vector space models.

The same applies to Cut reasoning. On the other hand, XML vector space retrieval has a monotonic reasoning behaviour that is in-between the ones for the flat vector space model and the pure type vector space. We shall see in Chapter 8 how this is to its advantage, when we analyse the experimental evaluation using our theoretical insights.

Although the conditional support for LMU and RMU helps with the experimental performance, Mix and Context-Free And are not fully supported by XML vector space retrieval. We have seen that the conditional support for these two reasoning properties is a direct consequence of the fact that the conditions to the monotonic reasoning are external to the aboutness decision. This is in contrast to pure type XML retrieval. It is finally interesting that Containment and its monotonicity reasoning are both not supported by the XML vector space model, which marks a difference to its flat equivalent and makes its reasoning behaviour closer to pure type XML retrieval and its support of structural reasoning. In all the applicable preclusion-induced reasoning, we also see a behaviour closer to pure type XML retrieval than to the flat vector space retrieval. We come back to this when explain experimental results in Chapter 8.

In the next section, we investigate another type of a successful flat document retrieval model that has been adjusted to XML retrieval. For this purpose, we present two language modelling approaches.

5.3 Language Models

Language models have proven to be a popular method for XML retrieval. We will soon see in more detail why this is the case. For the time being, it suffices to say that they introduce an easy way of changing the unit for indexing. In XML retrieval different language models are calculated, one per document component. Parents and children are then related by combining their document component language models. We concentrate on two XML retrieval models [Sigurbjörnsson and Kamps, 2005] and [Ogilvie and Callan, 2005]. We shall call the first one XML Language Modelling I and the second one XML Language Modelling II.

5.3.1 Background

Language models have been introduced in [Ponte and Croft, 1998]. Their research has proven that language modelling is a powerful and flexible tool to provide solutions to numerous problems in IR from ad-hoc information retrieval to modern web retrieval.

[Ponte and Croft, 1998]'s idea for language modelling was to measure which terms t will probably be asked for when searching for a document d.

$$P(Q|M_d) = \prod_{t \in Q} p(t|M_d) \times \prod_{t \notin Q} (1.0 - p(t|M_d))$$

According to this formula based on the maximum likelihood estimate from [Ponte and Croft, 1998] the language modelling ranking depends on the combined probabilities of producing the terms t of a query Q and not producing other terms from the language model M_d of a document.

The approach is based on the modelling of the languages in documents and queries. The retrieval task is under these circumstances to generate the query as a random process from the documents' language models. Hence, a language model for each document has to be inferred and the probability of a query given that document has to be estimated.

Language models are part of the probabilistic approaches to IR. These have been seen to be extremely powerful and elegant [van Rijsbergen, 2000]. The main argument is that probabilistic methods best express the imperfect knowledge underlying the decision whether a document component is about a query.

According to our methodology from Section 4.1, we need a flat document equivalent for language models first, before we can continue with our examination of XML retrieval approaches. We (briefly) present the aboutness decision in the flat document language model based on a commonly used simplification of [Ponte and Croft, 1998].

5.3.2 Theoretical Evaluation of Flat Document Language Modelling

For our flat document equivalent we use the standard version of language modelling as presented in [Manning et al., 2008]. This is a simplified version of [Ponte and Croft, 1998], but it is the underlying model of many variations including those used in XML retrieval, as we show later. In the model, words are determined that most likely appear in a relevant document. The documents are then ranked based on the probability whether they contain the query words often.

In order to rank documents, we first need to infer a language model for a document and estimate the probability of generating the query from there. In [Manning et al., 2008], the language model of any document is based on the maximum likelihood estimate, which estimates the language model of a document by considering how many times a term occurs in the document: $P(q|M_d) = \Pi_{t \in q} \frac{t_{f,d}}{t_d}$, where $t_{f,d}$ is the frequency of a term t in a document d and t_d is the overall number of terms in the document. M_d is the language model and q the query.

As only the terms that are part of the query are considered, we suspect that the model is also built on information overlap. This would correspond to our experience from the other simple model based on information overlap, the plain vector space model from Section 4.6. Furthermore, it seems that aboutness is also simply given by $P(q|M_d) > 0$. No external threshold is defined in the model.

However, looking at the example calculations in Section 5.3.3.2, we see that $P(q|M_d)$ is never 0. This is the case, because smoothing takes place to mitigate the problem that terms only appear sparsely in documents. Without smoothing, $P(q|M_d)$ would only be larger than 0 if all query terms appear in the document. This is too strict. We need to smooth the probabilities in a language model and discount non-zero probabilities.

There are many smoothing approaches according to [Manning et al., 2008]. We choose the one that is also used in the XML language modelling approaches we analyse. It is called the linear interpolation approach or Jelinek-Mercer approach [Manning et al., 2008]. Here: $P(t|M_d) = \lambda P(t|M_d) + (1 - \lambda)P(t|M_c)$. λ is a tuning constant between 0 and 1, which we can ignore in our discussions. M_c is the collection language model of a term t and the smoothing value. This means that $P(q|M_d) = \Pi_{t \in q}(\lambda P(t|M_d) + (1 - \lambda)P(t|M_c))$. As we only consider terms that are part of the collection, $(1 - \lambda)P(t|M_c) > 0$. Then also, $P(q|M_d) > 0$.

According to the Jelinek-Mercer approach, the smoothing value is the smallest value for estimating how much a term contributes to a language model. Then, the smallest possible value for $P(q|M_d)$ will be a product of smoothing values if we ignore the constant λ. In fact, it will be the product of smoothing values for all terms in the collection c: $\prod_{t \in c}(1 - \lambda)P(t|M_c)$. We call this value θ. It describes an internal threshold and is the retrieval status value of a document that has all collection terms but no query terms. The document is therefore not able to generate the query language model, because the query asks for information that cannot be found in the document collection.

[Manning et al., 2008] also discuss that the value of smoothing goes beyond a technical correction to avoid problems with sparsely distributed terms t. It has an impact on the performance of the model as an internal threshold of aboutness, as we shall see.

5.3.2.1 Translation

In [Manning et al., 2008], the document information is captured as terms used to generate the language models. We are able to reuse parts of our vector space model translation from Section 4.6.2. In this case $\chi(d)$ describes the set of descriptors (again with index terms and other information to calculate rsv), while $\chi(q)$ corresponds to the descriptors in q. We again use the basic infon language, as defined on page 42. Index term are directly translated into infons and the set of all index terms infons is the document situation.

$$map(\chi(d)) = \{ \langle\langle Value, t; 1\rangle\rangle \,|t \in \chi(d)\}$$

The operators equivalence, composition, containment and preclusion are defined analogously to the ones for the simple vector space model.

- Equivalence: Given two situations S and T, $S \equiv T =^{def} (\varphi \in S \Leftrightarrow \varphi \in T)$, where φ is any infon based on all keywords in the document collection. This means that the underlying language models for S and T are the same, which implies in our case that they contain the same descriptors from a collection.
- Composition: Given two situations S and T, $S \otimes T =^{def} (S \cup T)^{(p_1,r_1,\ldots,p_n,r_n)(q_1,s_1,\ldots,q_n,s_n)}$ with p,q and r,s are parameters used in S and T respectively. The resulting new language model using the composition operator is simply the combined set of descriptors from a collection.
- Containment: Given two situations S and T, $S \rightarrow T =^{def} (\varphi \in S \rightarrow \varphi \in T)$, where φ is any infon based on all descriptors in the document collection. In terms of language models, this means that S's descriptors can all be found in T, too, and T has no other descriptors.
- Preclusion is not applicable beyond simple non-aboutness, similarly to what we said about the simple vector space model in Section 4.6.2.

The next step presents the rules for flat language modelling in order to process afterwards the theoretical evaluation for the XML retrieval models.

5.3.2.2 Rules

Next, we introduce the *flat document language modelling aboutness decision*, based on [Manning et al., 2008]. Let $\mathcal{D}$ be a set of documents and d be a document in it. q represents a query and θ the internal threshold described above. The language modelling aboutness decision is:

$$d \text{ about } q \text{ if and if only } P(q|M_d) > \theta$$

Then, our aboutness definition states that a document represented by $\chi(d)$ is about a query $(\chi(q))$ if the set of index terms that constitute the document's language model induces a large enough belief into the language model of the query. The large enough belief is measured by θ, which is smallest possible value for rsv based on the product of smoothing values.

We try and base our aboutness system on our existing analyses of other aboutness systems, which is in this case the simple (non-thresholded) vector space retrieval model from Section 4.6. This way we can avoid having to go through all the rules. As explained in Section 5.3.2, the language model is essentially built on information overlap (measured in descriptors' overlap) between document and query. We therefore claim that the following Proposition holds with given sets of descriptors $\chi(d)$ describing a document d and $\chi(q)$ describing a query q:

Proposition 5.3.1 $P(q|M_d) > \theta \Leftrightarrow \chi(d) \cap \chi(q) \not\equiv \emptyset$

Proposition 5.3.1 can be proven by:

Proof $\Rightarrow$: Assume $P(q|M_d) > \theta$. This means also that $P(t|M_d) > 0$ for some terms t. As $t_d > 0$ according to its definition, this must then also imply that $t_{f,d} > 0$, which counts in $P(q|M_d)$ the number of times an index term from a query appears in a document. As $t_{f,d} > 0$, there has to be at least one descriptor, which is part of both document and query or $\chi(d) \cap \chi(q) \not\equiv \emptyset$.
$\Leftarrow$: Assume $\chi(d) \cap \chi(q) \not\equiv \emptyset$, which implies there is at least one descriptor part of d and q. Thus, $t_{f,d} > 0$ and $t_d > 0$ and finally $P(q|M_d) > \theta$. ∎

Proposition 5.3.1 shows that the aboutness decision for language models is based on information overlap between query and document, which determines whether $P(Q|M_d) > \theta$. This looks similar to the thresholded vector space model, for which we also have the same basic infon language. However, the decisive difference is that the threshold in the vector space model functions as a means to control aboutness behaviour, while here it is mainly used to avoid undesired side effects in P. Looking at Proposition 5.3.1 the aboutness decision of the flat language model seems to be more related to the simple vector space model one from Section 4.6. Both models are embedded in each other according to Huibers' definition [Huibers, 1996]. Thus, for the analysis of language modelling, we can focus on those rules that are either conditionally or fully supported by the simple vector space model from Section 4.6.

Next, we prove the reasoning properties using Proposition 5.3.1. We can keep the proofs very brief, as they are similar to the ones for the simple vector space model. As for the vector space retrieval, we would like to exclude Reflexivity in order to avoid logical anomalities. **Singleton Reflexivity** is supported for simple language modelling. Say, that A and B are both sets of descriptors. We have to show that, with $map(A) \equiv \{\phi\}$ and $map(B) \equiv \{\phi\}$, then $A \cap B \not\equiv \emptyset$. The latter is the case as ϕ is part of both. Singleton Reflexivity is given (according to the proposition).

Symmetry is supported, too. Say $S \equiv map(A)$ and $T \equiv map(B)$. Symmetry is given, as $\cap$ in $A \cap B$ is commutative. Symmetry also clearly shows that the threshold θ does not control the aboutness behaviour. According to Section 5.3.2, we assume θ not to be a fixed value but the result of a function dependent on the number of descriptors in the collection. Because all terms in the collections are considered, θ will not change if we make the document the query, as implied by the Symmetry rule.

Set Equivalence is also supported. Then, $map(A) \equiv map(B)$ and $map(A) \square\!\!\leadsto map(C)$ are given according to the assumptions. We have to show that $map(B) \square\!\!\leadsto$

$map(C)$ is supported. From the premises, we know by the definition of map that $A \equiv B$ and $A \cap C \not\equiv \emptyset$ which includes $B \cap C \not\equiv \emptyset$. This proves that the Set Equivalence rule is given. We can omit those simple rules that were not supported for the plain vector space aboutness such as Transitivity, etc.

Regarding the combination rules, we find again an identical behaviour to plain vector space retrieval. **Left Monotonic Union** is supported if $S \otimes U \Box\!\!\rightsquigarrow T$ given that $S \Box\!\!\rightsquigarrow T$. Say, that $S \equiv map(A)$, $T \equiv map(B)$ and $S \otimes U \equiv map(C)$. Then, $A \cap B \not\equiv \emptyset$, as S is about T, and $C \supseteq A$ by definition of map. Then, also $C \cap B \not\equiv \emptyset$ and Left Monotonic Union is supported.

For **Right Monotonic Union**, we say that from $S \Box\!\!\rightsquigarrow T$ also $S \Box\!\!\rightsquigarrow T \otimes U$. Again: $S \equiv map(A)$, $T \equiv map(B)$ and $T \otimes U \equiv map(C)$. $A \cap B \not\equiv \emptyset$, with S about T. $C \supseteq B$ follows from the definition of map. With $A \cap C \not\equiv \emptyset$, Right Monotonic Union is supported. **Cut** and **Right Weakening** are not given, as they are not supported for plain vector space retrieval. **Mix** is given, as Left Monotonic Union is, and **Context-Free And** is supported, as Right Monotonic Union is.

Deep containment is not defined for the flat document language model. Containment, Containment Composition, Absorption, Right Containment Monotonicity, Non-conflict-containment, Closed World Assumption and Containment Preclusion are all only supported for **surface containment** for the model. S contains T if all descriptors in A can also be found in B and no other, with $S \equiv map(A)$ and $T \equiv map(B)$. Then, obviously $A \cap B \not\equiv \emptyset$.

Absorption is given according to the definitions of composition and containment. **Right Containment Monotonicity** concludes $S \Box\!\!\rightsquigarrow U$ from the assumptions $S \Box\!\!\rightsquigarrow T$ and $T \rightarrow U$. This means all elements in T can also be found in U. This does not imply that the index terms that constitute an overlap between S and T can also be found in both S and U. Right Containment Monotonicity is not given.

As preclusion is not defined for the flat language model, Non-conflict-containment and Containment Preclusion are not applicable. All the non-aboutness rules are then not applicable, too: Mutual Preclusion, Guarded Left Monotonicity, Guarded Right Monotonicity, Qualified Left Monotonicity and Qualified Right Monotonicity. Language models cannot control or qualify their monotonic behaviour. Without preclusion, non-aboutness also does not make sense for flat language modelling.

In the following sections we use this flat language model to compare the behaviour of language models for structured document retrieval. We begin with the model from [Sigurbjörnsson and Kamps, 2005].

5.3.3 XML Language Models I

5.3.3.1 Background

[Sigurbjörnsson and Kamps, 2005] use language models for XML retrieval. They argue that not all elements are equally likely to be seen as satisfactory answers to an information need [Sigurbjörnsson and Kamps, 2005], and that too small elements should generally not be regarded as relevant answers. As in the XML vector space model, each XML element is indexed separately: one index for overlapping elements, one length-based one, one for elements frequently appearing in assessment sets (Qrel) and one for sections. The complete article is kept in another index:

- Overlapping element index: This index contains all elements.
- Length-based index: Only those elements are kept that have an average length of more than 25 terms.
- Qrel-based index: Only elements are indexed that have appeared relatively frequently in previous assessment sets. These are article, bdy, sec, ss1, ss2, p, ip1 and fig in the context of the INEX test collections.
- Section index: Only section elements are indexed.
- Article index: The complete article is indexed.
- Fielded index: The complete article is kept together with some selected fields for context restrictions in structured queries. In INEX 2005, they used the following restrictions most common to INEX 2003 and 2004 queries: abs, fm//au, fm//atl, kwd, st, bb//au, bb//atl, and ip1.

The ranking uses a variant of what we have described for the plain document language model:

$$P(q|e) = P(e) * \prod_{i=1}^{k} P(t_i|e)$$

q is a query with terms $t_1, ..., t_k$. e is an element. The language model is determined by interpolating element, document and collection language models:[1]

$$P(t_i|e) = \lambda_e * P_{mle}(t_i|e) + \lambda_d * P_{mle}(t_i|d) + (1 - \lambda_e - \lambda_d) * P_{mle}(t_i)$$

$P_{mle}(.|e)$ refers to the language model of an element e, $P_{mle}(.|d)$ of a document d containing e, and $P_{mle}(t_i)$ is a language model of the collection. Two training parameters are used, λ_e for the element model and λ_d for the document.

Table 5.11: dl_d

Q	D1	D2	D3	P_{11}	P_{12}	P_{21}	P_{22}	P_{31}	P_{32}
3	4	5	4	2	2	2	3	1	3

Table 5.12: $DocumentFrequency$

	house	garden	courtyard	garage	damage	fire	door	arrive
df_t	2	1	2	2	1	1	1	2

[1]Please note that for all discussed models we use their own notations.

We can easily identify the similarity to the plain document language model. In particular, the maximum likelihood estimate P_{mle} is again defined as the number of times a term occurs in a document component compared to the overall number of terms in that document component. We can thus assume that the model is again essentially using information overlap between document component and query to determine aboutness. Compared to Section 5.3.2, the linear interpolation smoothing approach is amended by the immediate context of the document component, the article language model. Please note that this interpolation does not relate a specific element to its direct context, to its children or ancestors, but only to its overall context, its document and its collections. This is important to later understand the way the model integrates structure. Finally, a length prior of an element e in a collection c is calculated:

$$P_c(e) = \frac{|e|}{\sum_{i \in c} |i|}$$

Next, we discuss the example.

5.3.3.2 Example

Table 5.13: tf

	house	garden	courtyard	garage	damage	fire	door	arrive
$D1$	1	0	0	1	1	1	0	0
$D2$	0	2	1	0	0	0	1	1
$D3$	1	0	1	1	0	0	0	1
P_{11}	1	0	1	0	0	0	0	0
P_{12}	0	0	0	0	1	1	0	0
P_{21}	0	1	0	0	0	0	1	0
P_{22}	0	1	1	0	0	0	0	1
P_{31}	1	0	0	0	0	0	0	0
P_{32}	0	0	1	1	0	0	0	1

We use the same example as in Section 5.2.2. Table 5.11 shows the number of term occurrences per document component.

Table 5.14: $P_{mle}(.|e)$

	house	garden	courtyard	garage	damage	fire	door	arrive
$D1$	0.25	0	0	0.25	0.25	0.25	0	0
$D2$	0	0.4	0.2	0	0	0	0.2	0.2
$D3$	0.25	0	0.25	0.25	0	0	0	0.25
P_{11}	0.5	0	0.5	0	0	0	0	0
P_{12}	0	0	0	0	0.5	0.5	0	0
P_{21}	0	0.5	0	0	0	0	0.5	0
P_{22}	0	0.33	0.33	0	0	0	0	0.33
P_{31}	1	0	0	0	0	0	0	0
P_{32}	0	0	0.33	0.33	0	0	0	0.33

In Table 5.12 we find the document frequency per term. Table 5.13 presents the term frequency per document component. The maximum likelihood is presented in Table 5.14.

Table 5.15: $P(t|C)$

house	garden	courtyard	garage	damage	fire	door	arrive
0.15	0.15	0.15	0.15	0.08	0.08	0.08	0.15

Again, the Jelinek-Mercer interpolation smoothing method is used. The results are shown in Table 5.15.

Table 5.16: Combinations

	house	garden	courtyard	garage	damage	fire	door	arrive
$D1$	0.19	0.09	0.09	0.19	0.15	0.15	0.05	0.09
$D2$	0.09	0.25	0.17	0.09	0.05	0.05	0.13	0.17
$D3$	0.19	0.09	0.19	0.19	0.05	0.05	0.05	0.19
P_{11}	0.22	0.09	0.09	0.22	0.12	0.12	0.05	0.09
P_{12}	0.17	0.09	0.09	0.17	0.17	0.17	0.05	0.09
P_{21}	0.09	0.26	0.15	0.09	0.05	0.05	0.22	0.11
P_{22}	0.09	0.23	0.21	0.09	0.05	0.05	0.07	0.21
P_{31}	0.42	0.09	0.12	0.12	0.05	0.05	0.05	0.12
P_{32}	0.12	0.09	0.21	0.21	0.05	0.05	0.05	0.21

Table 5.17: $P(e)$

D1	P_{11}	P_{12}	D2	P_{21}	P_{22}	D3	P_{31}	P_{32}
0.15	0.08	0.08	0.19	0.08	0.12	0.15	0.04	0.12

Following [Sigurbjörnsson and Kamps, 2005], we further assume $\lambda_e = 0.1$ and $\lambda_d = 0.3$. Then, we can calculate the interpolation of element, document and collection:

$$P(t_i|e) = \lambda_e * P_{mle}(t_i|e) + \lambda_d * P_{mle}(t_i|d) + (1 - \lambda_e - \lambda_d) * P_{mle}(t_i)$$

This leads to the combinations as represented in Table 5.16. Finally, the length prior $P(e) = \frac{|e|}{\sum_e |e|}$ is represented in Table 5.17.

The final probabilities are then: $P(q|D2) = 0.0007266$, $P(q|P_{22}) = 0.0005216$, $P(q|D3) = 0.0004874$, $P(q|P_{21}) = 0.0002808$, $P(q|P_{32}) = 0.0002722$, $P(q|D1) = 0.0002309$, $P(q|P_{31}) = 0.0001814$, $P(q|P_{11}) = 0.0001426$ and $P(q|P_{12}) = 0.0001102$.

Before continuing with the theoretical evaluation, it becomes clear by looking at the example that a combination of the smoothing values is the lowest possible value in Table 5.16. This means that no element, even though it has no information overlap with the query, will ever have a retrieval status value of 0, which has led us to a new form of threshold, which is a threshold that is not external as in the XML vector space model but internal.

5.3.3.3 Translation

The LM I model is similar to the XML vector space retrieval. Again, standard IR techniques are applied to XML retrieval by separating out the XML elements in the indexing step. The main difference is that the authors do not merge their retrieval results. Regarding the translation, the approach for the flat document language model has to be extended to reflect the separation into different indexes.

With respect to these initial considerations, the XML retrieval model LM I is generally built on the decision that a document d represented by a set of descriptors $\chi(d)$ is about a query q (represented by $\chi(q)$) if and if only the information in q can be found in the indexes below. We therefore have to give a *map* function to translate information items into situations for each of the above indexing methods.

1. $map_{full}(\chi(d)) = \{ \langle\langle ElementType, e, i; 1\rangle\rangle, \langle\langle Value, t, i; 1\rangle\rangle \, | t \in \chi(d)\}$.
2. $map_{length}(\chi(d)) = \{ \langle\langle ElementType, e, i; 1\rangle\rangle, \langle\langle Value, t, i; 1\rangle\rangle \, || \chi(d)| > \kappa\}$.
3. $map_{Qrel}(\chi(d)) = \{ \langle\langle ElementType, e, i; 1\rangle\rangle, \langle\langle Value, t, i; 1\rangle\rangle \, | e \in \{article, bdy, sec, ss1, ss2, p, ip1, fig\}, t \in \chi(d)\}$.
4. $map_{sec}(\chi(d)) = \{ \langle\langle ElementType, e, i; 1\rangle\rangle, \langle\langle Value, t, i; 1\rangle\rangle \, | e \in \{sec\}, t \in \chi(d)\}$.
5. $map_{article}(\chi(d)) = \{ \langle\langle ElementType, e, i; 1\rangle\rangle, \langle\langle Value, t, i; 1\rangle\rangle \, | e \in \{article\}, t \in \chi(d)\}$.
6. $map_{fielded}(\chi(d)) = \{ \langle\langle ElementType, e, i; 1\rangle\rangle, \langle\langle Value, t, i; 1\rangle\rangle \, | e \in \{abs, kwd, st, fm//au, fm//atl, bb//au, bb//atl, ip1\}, t \in \chi(d)\}$.

e is an element type in the set of all element types of a collection, t is a descriptor in the collection, i an identifier for infons and κ is a length threshold. Apart from the article index, the major difference to the flat language model is the division into document components instead of documents.

This translation appears similar to the one for the XML vector space model in Section 5.2.3, but for keeping separate indexes. In fact, we find a similar model repeated for almost all translations we encounter. Conceptually, however, this model is very different from the XML vector space model. Different indexes represent different experiences of the importance and impact of particular document components. This is 'hidden' in the map function which allows us to use straight-forward set operations.

Next we need to define the operators: equivalence, composition, containment and preclusion. We can reuse the ones for flat language modelling retrieval from Section 5.3.2.1 but need to consider now that for all operators we need to assume that they only relate situations, which are part of the same index. Then, given two situations S and T, they are equivalent if all their infons are identical and are part of the same index. S and T can be composed using $\cup$ and parameter replacement — again only within the same index. S contains T, if T has only infons from S. We thus define containment as surface containment. Deep containment is an addition to the model. Only preclusion differs slightly from what we have seen in 5.3.2.1. It is related to the preclusion for the XML vector space model from Section 5.2.3. A situation S describing a document component precludes another situation T if one of them is an element that is not part of conditions of a particular index. For instance, it is not part of the most informative elements or not a section or article or does not have the required length.

The next section investigates the aboutness rules.

5.3.3.4 Rules

Let $\mathcal{D}$ be a set of document components and d a document component and q a query. The *LM I XML language model aboutness decision* is then:

$$d \text{ about } q \text{ if and if only } P(q|M_d) > \theta$$

This formula looks like the flat language retrieval model one except for the interpolation, which includes the relationship of each element with its collection and its containing article. The internal threshold θ is still the product of the smoothing values for all terms in the collection. The second major change is that d stands for document components instead of documents. This similarity to the flat language model allows us to focus on those rules that have been implemented by the flat language model and look at how they change their behaviour for the different translations given above.

As the retrieval status value will only be θ, if any descriptor overlap between the language model of query and document component is excluded, and is again dependent on the collection language model plus this time the article language model, we can reuse Proposition 5.3.1: We say for two situations S and T and sets of descriptors A and B that $S \,\Box\!\!\rightsquigarrow\, T \Leftrightarrow map(A) \cap map(B) \not\equiv \emptyset$ for $S \equiv map(A)$ and $T \equiv map(B)$. We use $map(A) \cap map(B)$ instead of $A \cap B$ to reflect that we can only meaningfully combine elements in the same index. The proof is analogous to the one for the flat language model, as the maximum likelihood functions are the same. We do not need to repeat it here. This time we would need to argue that according to rsv an element e is about a query q if they both contain at least one common t_i.

The model is therefore embedded in the flat language model. We can focus on its properties, but the discussion has to reflect the collection- and article-based internal threshold and the introduction of structure by using different indexes. We can see here that our approach is different from the one by Huibers. In [Huibers, 1996], the focus is on proving the soundness of aboutness systems. We are more interested in the actual behaviour of XML retrieval models. For our inquiry, the formulation of the threshold θ and the interpolation of XML elements with their neighbouring XML elements are key. This also means that we cannot just rely on the proofs from Section 5.3.2.2 but have to start the discussion again.

Singleton Reflexivity is given for LM I. Let $map(A) \equiv \{\phi\}$ and $map(B) \equiv \{\phi\}$, then $map(A) \cap map(B) \not\equiv \emptyset$. The latter is the case as ϕ is part of both. Singleton Reflexivity is given.

Symmetry is a similar case. It is supported for all translations, too, for similar reasons as it has been for flat language models, too. **Set Equivalence** is supported because it is in the flat language model. Let us say that $S \equiv map(A)$, $T \equiv map(B)$ and $U \equiv map(C)$. Then, with $map(A) \equiv map(B)$ and $map(A) \,\Box\!\!\rightsquigarrow\, map(C)$, also $map(B) \,\Box\!\!\rightsquigarrow\, map(C)$. According to the definition of map we know that $S \equiv T$ means that they both are part of the same index and the same content. Thus, $map(B) \cap map(C) \not\equiv \emptyset$, which demonstrates that Set Equivalence is given. The remaining simple aboutness rules do not apply.

Next, let us have a look at the combination rules. Regarding **Left Monotonic Union**

(LMU), say that $S \equiv map(A)$, $T \equiv map(B)$ and $S \otimes U \equiv map(C)$. According to our assumptions, we have $map(A) \cap map(B) \not\equiv \emptyset$ and $map(C) \supseteq map(A)$. Thus, $map(C) \cap map(B) \not\equiv \emptyset$, and LMU is unconditionally supported. LMU is even fully given, if the element remains unchanged but the collection gets extended. Yet, the monotonic extension can only take place within the same index. We cannot add an element from a different index, as this would involve a completely changed aboutness decision. If, e.g., an article element is added to a section, the section translation could not be used for the aboutness decision anymore. Therefore, LMU is unconditionally given as long as we add information only in the same index.

Looking at **Right Monotonic Union**, we define $S \equiv map(A)$, $T \equiv map(B)$ and $T \otimes U \equiv map(C)$. If Right Monotonic Union would be given, then with $S \Box\!\!\rightsquigarrow T$ also $S \Box\!\!\rightsquigarrow T \otimes U$. We therefore have $map(A) \cap map(B) \not\equiv \emptyset$ and $map(C) \supseteq map(B)$. Thus, $map(A) \cap map(C) \not\equiv \emptyset$, and Right Monotonic Union is supported. The same limitations apply as for Left Monotonic Union.

Mix is a special case of Left Monotonic Union and is therefore also supported. Similarly to Left Monotonic Union, only those situations can be combined, which are part of the same index. This is interesting, as for XML retrieval parents and children are about the same queries and Mix should therefore be an automatic property, because it extends Left Monotonic Union. However, this is not the case for this model, where children and parents can be part of distinct indexes. **Context-Free And** holds, because Right Monotonic Union does. The conditions are the same as for Mix.

Only **surface containment** is applicable to the model. If there is an overlap in index terms of two subsituations, then their corresponding situations will be about each other. As a subsituation is surface-contained in its situation, **Absorption** also holds. Finally, **Right Containment Monotonicity** is not given, as it was not given for the flat language model.

Preclusion is not defined for the LM I XML retrieval model. Therefore, all rules involving preclusion are not applicable: Non-conflict-containment, Containment Preclusion, Mutual Preclusion, Guarded Left Monotonicity, Guarded Right Monotonicity, Qualified Left Monotonicity and Qualified Right Monotonicity. The Closed World Assumption does not apply.

5.3.3.5 Completeness

To demonstrate completeness of the above rules, we have to show that for two descriptor sets A and B (from the same index) if A contains descriptors from B, then $map(A) \Box\!\!\rightsquigarrow map(B)$. Let us say that $C \equiv A \cap B$. C is the set of descriptors A and B have in common. Let us assume that $D \equiv A \setminus C$ and $E \equiv B \setminus C$. Furthermore, $S \equiv map(A)$, $T \equiv map(B)$, $U \equiv map(C)$, $V \equiv map(D)$ and $W \equiv map(E)$. As our aboutness proof system includes the rules Left Monotonic Union, Right Monotonic Union and Set Equivalence we can make the following conclusions. We begin with Reflexivity and state:

$$U \Box\!\!\rightsquigarrow U$$

With Right Monotonic Union, we can then also say:

$$\frac{U \square\!\!\rightsquigarrow U}{U \square\!\!\rightsquigarrow U \otimes W}$$

Using Set Equivalence, we can derive:

$$\frac{U \square\!\!\rightsquigarrow U \otimes W, T \equiv U \otimes W}{U \square\!\!\rightsquigarrow T}$$

Using an analogue combination of Set Equivalence and Left Monotonic Union finally delivers:

$$\frac{U \square\!\!\rightsquigarrow T}{S \square\!\!\rightsquigarrow T}$$

Therefore, the aboutness proof system has to be complete, as we have all the rules needed to conclude that S is about T. Please note that we have stayed within the same index. It is obvious that Completeness holds for all indexes.

5.3.3.6 Reflection

As demonstrated in Section 4.5, the reflection step determines those document components and queries that are top or bottom exhaustive or specific.

1. A bottom exhaustive document component D_j is never exhaustively about any query Q: $\{D_j | D_j \in \mathcal{D}, D_j \square\!\!\not\rightsquigarrow Q\}$. $\{\emptyset\}$ is D_j. The only D that is never exhaustively about Q is $\{\emptyset\}$, as $\emptyset \cap \emptyset \equiv \emptyset$. Next, we assume that D is any other situation S with $S \not\equiv \{\emptyset\}$. But then: $S \cap \emptyset \not\equiv \emptyset$. Thus, D is always only $\{\emptyset\}$. The value of its language model is the interpolation of document and collection language model.

2. A bottom exhaustive query Q_j is the one in which all document components D are never exhaustive answers to: $\{Q_j | Q_j \in \mathcal{Q}, D \square\!\!\not\rightsquigarrow Q_j\}$. The bottom exhaustive query is $\{\emptyset\}$. Let us assume the document component D is itself $\{\emptyset\}$. Then, the only query Q that any D is never an exhaustive answer to is $\{\emptyset\}$. Let us assume $D \not\equiv \{\emptyset\}$. Then, with $D \equiv map(A)$ the only never given overlap to $map(A)$ is $\{\emptyset\}$. Therefore, $\{\emptyset\}$ is the bottom exhaustive query. Its language model will again be the interpolation of collection and document model.

3. A bottom specific document component D_j is never specifically about any query Q: $\{D_j | D_j \in \mathcal{D}, Q \square\!\!\not\rightsquigarrow D_j\}$. The bottom specific document component is $\{\emptyset\}$. The proof is analogous to the one for bottom exhaustive queries.

4. A bottom specific query Q_j is the one to which all document components D are never specific answers: $\{Q_j | Q_j \in \mathcal{Q}, Q_j \square\!\!\not\rightsquigarrow D\}$. The bottom specific query is again $\{\emptyset\}$.

As we found bottom exhaustive document component and query situations, we do not have top exhaustive document component and query situations. Also, having a bottom

Table 5.18: Reflection of structural behaviour

	LM I XML Language Model	Pure type XML retrieval
Top Exhaustive Document Component		(virtual) root
Top Exhaustive Query		$\{\emptyset\}$
Bottom Exhaustive Document Component	$\{\emptyset\}$	
Bottom Exhaustive Query	$\{\emptyset\}$	
Top Specific Document Component		$\{\emptyset\}$
Top Specific Query		(virtual) root
Bottom Specific Document Component	$\{\emptyset\}$	
Bottom Specific Query	$\{\emptyset\}$	

specific document component and query situation, we can exclude top specific document component and query situations.

Like the XML vector space retrieval model, this model is distinctively different from our pure type XML retrieval model. In fact, its reflection shows identical behaviour to the XML vector space retrieval model, as both are built on information overlap. As in Section 5.2 for the vector space model, we see no way to incorporate structure in the aboutness decision. Structure is included in the aboutness decision by a priori dividing elements into several different indexes. [Sigurbjörnsson and Kamps, 2005] note as one of the main complications with their approach, that they

> 'are using widely different indexes, varying from an index containing all individual elements or subtrees to indexes containing only the article or section elements.'

Therefore, they conclude that 'it is non-trivial to compare [...] over different indexes.' In the next section we look at a second language modelling approach that has found a way to include XML structure in the aboutness reasoning.

5.3.4 XML Language Modelling II

5.3.4.1 Background

Another language modelling approach to XML retrieval has been presented in INEX 2004 [Ogilvie and Callan, 2004] and INEX 2005 [Ogilvie and Callan, 2005]. Documents are modelled using a tree-based language model, which estimates the probability of the query using the document compoments language models. Apart from a different notation, the formula is again very similar to the flat document retrieval one in Section 5.3.2. Yet, this time each language model (here called μ_e) is estimated by using evidence from the document, its parent and the children:

$$\begin{aligned}P(w|\mu_e) &= \lambda_P(w|\theta_{P(e)}) + \lambda_D(w|\theta_{D(e)}) + \lambda_C(w|\theta_C) \\ &\quad + \lambda_O \frac{|s(e)|}{|s(e)| + \sum_{j' \in c(e)} \alpha_{t(j')}|j'|} P(w|\theta_{s(e)}) \\ &\quad + \lambda_O \sum_{j \in c(e)} \frac{\alpha_{t(j')}|j|}{|s(e)| + \sum_{j' \in c(j)} \alpha_{t(j')}|j'|} P(w|\theta_j) \qquad (5.3.1)\end{aligned}$$

θ_x is the language model estimated for x, $P(x)$ is the parent of x, $D(x)$ the document containing x, $s(x)$ the element x, $c(x)$ returns a list containing the children of x, $t(x)$ is the element type of the element x and C refers to the entire collection. The λ parameters in the interpolation are set to be constant across all elements in the collection and estimated beforehand using a training data set. The α parameters allow to provide additional weights to particular types of children of elements.

The same formula is used again for the maximum likelihood estimation describing the language model of each XML element:

$$P(w|\theta_x) = \frac{freq(w,x)}{|x|}$$

x is the observed text in the XML elements, $freq(w,x)$ is the number of times the term w occurs in x, and $|x|$ is the length in terms of x.

Rankable items are finally ordered by:

$$P(Q|\mu_e) = \prod_{i=1}^{|Q|} P(q_i|\mu_e)$$

μ_e is the language model estimated for a particular element e. Finally, a linear length prior is applied by multiplying the length of a relevant element with $P(Q|\mu)$.

Table 5.19: $\lambda_C(w|\theta_C)$

house	garden	courtyard	garage	damage	fire	door	arrive
0.07	0.07	0.07	0.07	0.04	0.04	0.04	0.07

Table 5.20: $\lambda_D(w|\theta_{D(e)})$

	house	garden	courtyard	garage	damage	fire	door	arrive
$D1$	0.06	0	0	0.06	0.06	0.06	0	0
$D2$	0	0.09	0.04	0	0	0	0.04	0.04
$D3$	0.06	0	0.06	0.06	0	0	0	0.06
P_{11}	0.06	0	0	0.06	0.06	0.06	0	0
P_{12}	0.06	0	0	0.06	0.06	0.06	0	0
P_{21}	0	0.09	0.04	0	0	0	0.04	0.04
P_{22}	0	0.09	0.04	0	0	0	0.04	0.04
P_{31}	0.06	0	0.06	0.06	0	0	0	0.06
P_{32}	0.06	0	0.06	0.06	0	0	0	0.06

Next we go through an example to show how the model works.

5.3.4.2 Example

Table 5.21: $\lambda_P(w|\theta_{P(e)})$

	house	garden	courtyard	garage	damage	fire	door	arrive
$D1$	0	0	0	0	0	0	0	0
$D2$	0	0	0	0	0	0	0	0
$D3$	0	0	0	0	0	0	0	0
P_{11}	0.01	0	0	0.01	0.01	0.01	0	0
P_{12}	0	0.01	0.01	0	0	0	0.01	0.01
P_{21}	0	0.02	0	0	0	0	0.02	0
P_{22}	0	0.01	0.01	0	0	0	0	0.01
P_{31}	0.04	0	0	0	0	0	0	0
P_{32}	0	0	0.01	0.01	0	0	0	0.01

We use the same example, which we have already used for the LM I and the XML vector space retrieval model (Section 5.2.2). Then, the maximum likelihood estimate is the same as in Table 5.14. As linear priors, we use those which performed best in the experiments in [Ogilvie and Callan, 2005]: $\lambda_C = 0.475$, $\lambda_D = 0.222$, $\lambda_P = 0.035$ and $\lambda_O = 0.268$, while α is in our case 0.23, as we only have paragraph elements in the example. Tables 5.19 to 5.24 present the calculations.

Table 5.22: $\lambda_O \frac{|s(e)|}{|s(e)|+\sum_{j'\in c(e)} \alpha_{t(j')}|j'|} P(w|\theta_{s(e)})$

	house	garden	courtyard	garage	damage	fire	door	arrive
$D1$	0.06	0	0	0.06	0.06	0.06	0	0
$D2$	0	0.09	0.04	0	0	0	0.04	0.04
$D3$	0.06	0	0.06	0.06	0	0	0	0.06
P_{11}	0.11	0	0.11	0	0	0	0	0
P_{12}	0	0	0	0	0.11	0.11	0	0
P_{21}	0	0.11	0	0	0	0	0.11	0
P_{22}	0	0.07	0.07	0	0	0	0	0.07
P_{31}	0.22	0	0	0	0	0	0	0
P_{32}	0	0	0.07	0.07	0	0	0	0.07

Table 5.23: $\lambda_O \frac{|s(e)|}{|s(e)|+\sum_{j'\in c(e)} \alpha_{t(j')}|j'|} P(w|\theta_{s(e)})$

	house	garden	courtyard	garage	damage	fire	door	arrive
$D1$	0.02	0	0	0.02	0.02	0.02	0	0
$D2$	0	0.03	0.01	0	0	0	0.01	0.01
$D3$	0.02	0	0.02	0.02	0	0	0	0.02
P_{11}	0	0	0	0	0	0	0	0
P_{12}	0	0	0	0	0	0	0	0
P_{21}	0	0	0	0	0	0	0	0
P_{22}	0	0	0	0	0	0	0	0
P_{31}	0	0	0	0	0	0	0	0
P_{32}	0	0	0	0	0	0	0	0

Overall, the ranking is calculated using: $P(Q|\mu_e) = \prod_{i=1}^{|Q|} P(q_i|\mu_e)$. The results are (including the length prior): $P(Q|D2) = 0.01568$, $P(Q|P_{22}) = 0.010584$, $P(Q|D3) = 0.012348$,

Table 5.24: $P(w|\mu_e)$

	house	garden	courtyard	garage	damage	fire	door	arrive
$D1$	0.21	0.07	0.07	0.21	0.18	0.18	0.04	0.07
$D2$	0.07	0.28	0.16	0.07	0.04	0.04	0.13	0.16
$D3$	0.21	0.07	0.21	0.21	0.04	0.04	0.04	0.21
P_{11}	0.25	0.07	0.18	0.14	0.11	0.11	0.04	0.07
P_{12}	0.13	0.08	0.08	0.13	0.21	0.21	0.05	0.08
P_{21}	0.07	0.29	0.11	0.07	0.04	0.04	0.21	0.11
P_{22}	0.07	0.24	0.21	0.07	0.04	0.04	0.08	0.19
P_{31}	0.39	0.07	0.13	0.13	0.04	0.04	0.04	0.13
P_{32}	0.13	0.07	0.21	0.21	0.04	0.04	0.04	0.21

$P(Q|P_{11}) = 0.0063$, $P(Q|P_{32}) = 0.005733$, $P(Q|P_{21}) = 0.004466$, $P(Q|D1) = 0.004116$, $P(Q|P_{31}) = 0.003549$ and $P(Q|P_{12}) = 0.001664$.

5.3.4.3 Translation

Regarding the translation, the model keeps the XML structure during indexing in order to relate elements to each other in the aboutness reasoning. As the XML structure is kept, we can reuse how, in general, XML trees can be translated into situations from Section 4.7.2 and reuse its translation with new descriptors that contain the index terms as well as their frequencies in a document component and the number of terms in that document component. This is the case because LM II considers each XML element to be a combination of its own language model with the language model of its parents and children.[1] We can also reuse the operators from pure type XML retrieval. Two situations S and T are equivalent if they contain the same infons with parameter exchange. They can be combined using $\cup$, while preclusion and containment are defined using element type relationships.

The main difference between the Language Modelling I and Language Modelling II as well as the XML vector space retrieval model is that structure is not just represented during indexing but during the calculation of the retrieval status value. This can be done given that for XML documents structurally related elements are also content related, as we have seen in Section 5.2. Therefore, the language model of an element can be considered to be dependent on the language model of its relatives — in the model by combining an element's and its descendents' language models.

Each document component is considered to be a separate indexing unit. Structure comes into play during the actual ranking while calculating the language models, not only prior to the relevance calculation as in XML vector space and LM I. Hence, we have to take into consideration structural constraints, while analysing the reasoning as represented in the aboutness rules.

[1] Please note that only the *Parent* relation is represented, while attributes are not defined.

5.3.4.4 Rules

In order to decide whether an XML element d is about a query q the aboutness decision for the LM II model is defined as follows:

$$d \text{ about } q \text{ if and if only } P(q|\mu_e) > \theta$$

As in the LM I model, θ is an internal condition. It is the background model retrieval status value of an element. According to Tables 5.19 to 5.23, the smallest possible retrieval status value is found in document components that are part of documents which contain no query terms. One can clearly see this in Table 5.24. Just like in the flat document retrieval model, θ is therefore the product of the smoothing values for all terms in the collection.

We continue with examining which rules are included in the model. We can again focus on those rules that were already supported by flat language modelling, as the LM II model is an extension to it. The first rule to discuss is Reflexivity. **Singleton Reflexivity** cannot be extended to Reflexivity in general to include calculations over empty sets, as this can lead to undefined probability calculation. Otherwise, we would be in danger of undefined divisions by zero [Ponte and Croft, 1998].

The question is whether for Singleton Reflexivity with $map(A) \equiv \{\psi\}$ we can derive $P_{LM}(q|\mu_e) > \theta$. This has to be the case. We can ignore parent and children retrieval status values, as we only consider a singleton element. $P(q|\mu_e)$ is larger than θ as the language representation of ψ is about itself. Therefore, Singleton Reflexivity holds.

We could now continue with a detailed analysis of the rules and would soon discover that the model does not differ from LM I in the rules it supports. It only differs in the conditions. We therefore would just like to focus on a discussion of the monotonic behaviour and the impact of θ on it.

Left Monotonic Union would be a property of the aboutness systems, if with $S \,\Box\!\!\rightsquigarrow T$ we could derive that $S \otimes U \,\Box\!\!\rightsquigarrow T$. The question we need to answer is whether the extension to $S \otimes U$ will ever result in $S \otimes U \,\Box\!\!\not\rightsquigarrow T$. This is not the case. We need to distinguish between three cases:

1. S is extended with new content. This means that in Equation (5.3.1) the maximum likelihood estimate might increase if new relevant terms are added. As, however, in Equation (5.3.1) the existing set of relevant terms will not change, the retrieval status value can only ever increase. The threshold is always passed.

2. U is the parent of S. Whatever the retrieval status value of this parent, we add it in the parent part of Equation (5.3.1), which again means it cannot be reduced.

3. U is a child of S. Again, we only add retrieval status values of children without changing S.

Thus, Left Monotonic Union is unconditionally supported.

In order to offer better control of monotonic behaviour, an improvement of the model would be to introduce a threshold that would exclude elements with low retrieval status values such as P_{12} in the example from Section 5.3.4.2. However, it is difficult to describe a threshold a priori. Furthermore, the inclusion of negative examples of irrelevant elements seemed to have improved the model according to [Ogilvie and Callan, 2005]. This will again influence the monotonic behaviour, as we show in Section 5.4.2, where we discuss a similar strategy for another model. Another suggestion could be to put more emphasis on XML structure. The approach in [Ogilvie and Callan, 2005] is still limited to unstructured queries.

All XML retrieval approaches presented so far are not able to accommodate structure without more fundamental changes to their models, as with them the queries can never be given the same representation as the documents. Both deliver structure only in terms of different indexes for different elements. As for the first time we have a translation that allows for the inclusion of structure, it becomes interesting at this point to briefly look at the potential impact of queries considering XML structure on the reasoning behaviour. The advantages of this model, from a theoretical point of view, stem from the fact that within its mathematisation it is possible to express structural constraints by using conditional probabilities based on the element types. Contrary to that, in LM I we also had interpolation of different XML elements, but never the direct context of an XML element and only the overall document and collection values.

[Ogilvie and Callan, 2005] go into great detail to explain that in future work their model would be able to accommodate the importance of different XML element types for estimating the relevance of document components. They claim that this would include XML structure in the aboutness relation. For the remainder of this section, we would like to investigate this claim.

In the LM II model, information items are not only simply keywords anymore but keywords bound to element types, where the latter are themselves part of particular XML subdocuments. We have the possibility to query for subdocuments by adding XML element type definitions to the query. Hence, d and q are now represented by a set of descriptors $\chi(d)$ and $\chi(q)$ so that the XML structure is preserved. We define an aboutness relation that includes XML structure as defined in the model, as a relationship of the language models of XML elements. We use $\succeq$ to state that one language model can be constructed from another one.

$$d \text{ about } q \text{ if and only if } \chi(d) \succeq \chi(q)$$

$\succeq$ refers to the fact that we can construct the XML document language model of q starting from d.

This time, we only discuss those reasoning properties we need for the completeness proof in Section 5.3.4.5. **Reflexivity** is obviously given, because of the final ranking formula in Section 5.3.4. It implies to only consider those terms t in the XML elements that are also part of the query. Let us assume this is the set of descriptors $\mathcal{T}$. One way of constructing a new language model out of the terms in S is just to copy these descriptors. Thus, Reflexivity is given. For similar reasons, **Set Equivalence** holds.

For similar reasons, **Left Monotonic Union** and **Mix** also hold. $\mathcal{T}$ is only set to grow if we either extend $S \Box\!\!\leadsto T$ to $S \otimes U \Box\!\!\leadsto T$ for Left Monotonic Union or $S \Box\!\!\leadsto$

T and $S \Box\!\!\rightsquigarrow U$ to $S \Box\!\!\rightsquigarrow T \otimes U$ for Mix. Constructing a new document language model from one that is already about a query always implies that we preserve the existing aboutness relation. Thus, Left Monotonic Union and Mix are given. The monotonic behaviour, induced by the behaviour of $\mathcal{T}$, also takes care that **Right Monotonic Union** and **Context-Free And** hold.

The analysis of Language Modelling II demonstrates clearly that the inclusion of XML-enhanced queries does not automatically lead to a closer consideration of XML structure. The model includes structure in its reasoning as a relation between the contents of relatives in an XML document, because the language models are defined over content relations only. This is very different from the pure type XML retrieval model from Section 4.7, where hierarchical inclusion as an aboutness decision means to consider structure as a condition of a content relation.

5.3.4.5 Completeness

Regarding the completeness proof for XML Language Modelling II, the one for unstructured queries is the same as for XML Language Modelling I from Section 5.3.3.5, except for changes in the translation.

For structured queries, we have to show that for the XML documents A and B: If $A \succeq B$ then $map(A) \Box\!\!\rightsquigarrow map(B)$. $A \succeq B$ is given if the language model of A can be constructed from B. Thus, we need to be able to decide $map(A) \Box\!\!\rightsquigarrow map(B)$ for each way of constructing a new language model. All the ways of creating the language model for A from the language model for B need to be covered by our derivation system. We just sketch the proof here. B has to be a non-empty language model, as otherwise A could not be constructed by it. First, we cover the case where A is the same language model as B. Then, both will have the same index terms as descriptors. This also means their corresponding situations will contain the same infons (with different parameters), which leads to $map(A) \Box\!\!\rightsquigarrow map(B)$ using Reflexivity and Set Equivalence. If A is constructed from B by adding new information (new index terms or new elements), we can prove that $map(A) \Box\!\!\rightsquigarrow map(B)$ with Reflexivity, LMU, RMU and Set Equivalence using the set of index terms that define $A \succeq B$.

5.3.4.6 Reflection

The reflection for the LM II model is the same as the one for the LM I model with similar problems as seen before.

5.3.5 Conclusion

Table 5.25 summarises the results of our theoretical evaluation for all our the language models. We can clearly see that XML language modelling is different from the reasoning behaviour of pure type XML retrieval. Contrary to the reasoning behaviour of XML vector space retrieval, exhibited by Table 5.10, we find no real thresholded aboutness behaviour for both language modelling approaches. Both language modelling approaches

Reasoning behaviour	*Plain LM*	*XML LM I*	*XML LM II*	*Pure Type*
Singleton Reflexivity	fully	fully	fully	N/A
Reflexivity	N/A	N/A	N/A	fully
Symmetry	fully	fully	fully	not
Set Equivalence	fully	fully	fully	fully
Transitivity	not	not	not	fully
Euclid	not	not	not	not
LMU	fully	fully	fully	fully
RMU	fully	fully	fully	not
Cut	not	not	not	fully
Right Weakening	not	not	not	not
Mix	fully	fully	fully	fully
Context-Free And	fully	fully	fully	fully
Containment	fully	fully	fully	not
Absorption	fully	fully	fully	fully
Right Containment Monotonicity	fully	fully	fully	not
Non-Conflict-Containment	N/A	N/A	N/A	fully
Containment Preclusion	N/A	N/A	N/A	fully
Mutual Preclusion	N/A	N/A	N/A	fully
Negation Rational	N/A	N/A	N/A	fully
Closed World Assumption	N/A	N/A	N/A	fully

Table 5.25: Language modelling retrieval evaluation results

show identical reasoning behaviour to their flat model equivalent and fail to add. Though there is an internal threshold, it does not help with advancing the structural reasoning capacities.

In the next sections, we look at models specifically designed for XML retrieval and how these include structure in their aboutness decision.

5.4 Structured Models

In this section, we investigate two models, which have been specifically designed to meet the challenges of XML retrieval. Both models use the XML structure of documents in their aboutness decisions. The first one, Gardens Point, has been among the most successful models presented at INEX.

5.4.1 Gardens Point XML Retrieval

5.4.1.1 Background

Gardens Point XML retrieval (GPX) is presented in [Geva, 2005], where five problems were identified when it comes to using standard IR approaches against XML document collections:

1. Adequate selection of elements that satisfy the query keywords' constraints.
2. Adequate selection of elements that satisfy the structural constraints.
3. The assignment of scores to elements with matching keywords and structures.
4. The propagation of scores to antecedent or descendant elements.
5. The selection of ranked lists of results for specific tasks.

We believe that GPX offers an interesting solution to steps 3 and 4 and makes it therefore different to the models we have investigated so far.

In [Geva, 2005], each XML element is identified by the XPath context. The model differentiates aboutness for leaf from aboutness for branch XML elements. Leaf elements are considered to be about the query if they contain at least one query term. A branch element is about a query if its subtree contains at least one leaf element that is about the query.

For leaf elements L:

$$rsv_L = K^{n-1} \sum_{i=1}^{n} \frac{t_i}{f_i} \tag{5.4.1}$$

n is the number of unique query terms. K^{n-1} supports those components with multiple distinct query terms, and $K > 1$. t_i is the frequency of the i-th query term in the component and f_i its collection frequency. Thus, the formula favours components with many unique query terms and penalises query terms frequent in the collection.

The weights of the leaf elements are propagated to form the weights for branch elements R:

$$rsv_R = D(c) \sum_{i=1}^{c} rsv_{L_i} \tag{5.4.2}$$

c stands for the number of relevant children elements. A decay factor $D(c)$ is used to control the propagation [Geva, 2005], where $D(c) = 0.49$ for $c = 1$ and $D(c) = 0.99$ otherwise. L_i is the relevance score of the i^{th} child element. Finally, the corresponding article score is added to each component to improve the performance of elements in highly relevant articles.

5.4.1.2 Example

Let us use again our example from Section 5.2.2. As a NEXI expression our query would be $//[about(., house, garden, courtyard)]$. Using $rsv_L = K^{n-1} \sum_{i=1}^{n} \frac{t_i}{f_i}$, we get the results in Table 5.26.

Table 5.26: L

	D1	D2	D3	P_{11}	P_{12}	P_{21}	P_{22}	P_{31}	P_{32}
L	0	0	0	0.5	0	0.5	0.5	0.5	0.5
n	1	2	2	1	0	1	2	1	1
t_1	1	0	1	1	0	0	0	1	0
t_2	0	1	0	0	0	1	1	0	0
t_3	0	1	1	0	0	0	1	0	1

The next calculation step is $rsv_R = D(n) \sum_{i=1}^{n} L_i$, which assigns values to the three branch elements $D1$, $D2$ and $D3$. The results are represented in Table 5.27.

Table 5.27: R

	D1	D2	D3	P_{11}	P_{12}	P_{21}	P_{22}	P_{31}	P_{32}
R	0.25	5.45	0.99	0.5	0	0.5	5	0.5	0.5

In the final step we add the article value to arrive at Table 5.28.

Table 5.28: Article Value

D1	D2	D3	P_{11}	P_{12}	P_{21}	P_{22}	P_{31}	P_{32}
0.5	10.9	1.98	0.75	0	5.95	10.45	1.49	1.49

The ranking is then $D2$, P_{22}, P_{21}, $D3$, P_{31}, P_{32}, P_{11}, $D1$, P_{12}.

The next section briefly considers the flat model equivalent in order to be able to proceed with our theoretical evaluation.

5.4.1.3 Flat Model Equivalent

For the flat model equivalent, we just use the calculation of the relevance scores in the leaf elements, as this is the equivalent of looking for content only without any consideration of the document structure. Let $\mathcal{D}$ be a set of documents and d be a document in it. q is a query. Both are represented by a simple bag of descriptors: $\chi(d)$ and $\chi(q)$. The *flat Gardens point retrieval aboutness decision* is then defined by

$$d \text{ about } q \text{ if and only if } rsv(\chi(d), \chi(q)) > 0$$

rsv is defined by the leaf element calculations: $rsv(\chi(d), \chi(q)) = K^{n-1} \sum_{i=1}^{n} \frac{t_i}{f_i}$.

For the translation, our standard basic infon language from Section 4.3 can be used:

$$map(\chi(d)) = \{ \langle\langle Value, t; 1\rangle\rangle \,|t \in \chi(d)\}$$

No surprises also in terms of the operators equivalence, composition, containment and preclusion. They are the same as in the plain vector space retrieval model from Section 4.6.

Singleton Reflexivity is given for the flat GPX model. With $map(A) \equiv \{\phi\}$ and $map(B) \equiv \{\phi\}$, $rsv(A, B) > 0$. Again, A and B are sets of descriptors. K^{n-1} in Equation (5.4.1) can never be 0 with $K > 0$. Therefore it does not influence the overall result. As $n = 1$, it is 1. Looking at the sum in rsv, it has to be 1 with $n = 1$ and $t_1 = 1$ as well as $f_1 = 1$. Singleton Reflexivity is given.

Transitivity is not supported. We can easily construct an example so that, with $S \equiv map(A)$, $T \equiv map(B)$ and $U \equiv map(C)$, $rsv(A, B) > 0$ and $rsv(B, C) > 0$ but not $rsv(A, C) > 0$. Transitivity is not supported. For similar reasons, **Euclid** is also not given.

If **Symmetry** would be given, then from $S \Box\!\!\rightsquigarrow T$ also $T \Box\!\!\rightsquigarrow S$. Say, $S \equiv map(A)$ and $T \equiv map(B)$. The t_i value in Equation (5.4.1) describes the overlap of descriptors in query and document component. This value does not change. If $\frac{t_i}{f_i} > 0$ for $S \Box\!\!\rightsquigarrow T$ then also $\frac{t_i}{f_i} > 0$ for $T \Box\!\!\rightsquigarrow S$. Symmetry is given.

Set Equivalence is also supported. Say, that $map(A) \equiv map(B)$ and $map(A) \Box\!\!\rightsquigarrow map(C)$ are given, we have to show that $map(B) \Box\!\!\rightsquigarrow map(C)$ is given. Exchanging in Equation (5.4.2) $t_1, ..., t_n$ or $f_1, ..., f_n$ with an equivalent set of terms $t'_1, ..., t'_n$ or $f'_1, ..., f'_n$ respectively does not change $rsv(A, B) > 0$. Set Equivalence is supported for both Left and Right Set Equivalence.

Regarding the combination rules, **Left Monotonic Union** is fully supported. With $S \Box\!\!\rightsquigarrow T$, $S \otimes U \Box\!\!\rightsquigarrow T$ is given. Say, that $S \equiv map(A)$, $T \equiv map(B)$ and $S \otimes U \equiv map(C)$. We need to distinguish two cases. The increased document component represented by S contains additional query descriptors. In Equation (5.4.1), the sum $\sum_{i=1}^{n} \frac{t_i}{f_i}$ is larger. If it does not contain additional query descriptors, it will not change. In both cases $rsv(A, C) > 0$ and Left Monotonic Union is supported.

Things are not different for **Right Monotonic Union**. The question is whether we can from $S \Box\!\!\rightsquigarrow T$ also say that $S \Box\!\!\rightsquigarrow T \otimes U$. If we add new term descriptors in T, n and also the sum in Equation (5.4.1) will increase potentially, but never decrease. Therefore, Right Monotonic Union is given.

Cut would allow to conclude $S \Box\!\!\rightsquigarrow T$, given that $D \otimes U \Box\!\!\rightsquigarrow T$ and $S \Box\!\!\rightsquigarrow U$. Clearly, we could eliminate all the occurrences of query descriptors in $S \otimes U$ without violating $S \Box\!\!\rightsquigarrow U$. Then, $n = 0$ in Equation (5.4.1), and Cut is not given. **Right Weakening** is not given either. We can take away all the query terms occurring in a document to make $n = 0$. Right Weakening does not hold. **Mix** will be supported as will be **Context-Free And** if Left Monotonic Union and Right Monotonic Union are supported.

Absorption is supported, but **Right Containment Monotonicity** does not hold: $S \Box\!\!\rightsquigarrow U$ does not necessarily hold if $S \Box\!\!\rightsquigarrow T$ and $T \rightarrow U$.

As preclusion is not defined for the model, Non-conflict-containment and Containment Preclusion are not applicable. As presented, then also all the non-aboutness rules using preclusion are not applicable. The **Closed World Assumption** is also not given. We continue with our theoretical evaluation of the full XML GPX model by defining the translation.

5.4.1.4 Translation

Let us assume that we have an XML document d for GPX retrieval. The translation function map is defined as follows, where we reuse the one from Section 4.7.2.1:

- For each XML element p with element type U in d, map is $\{ \langle\langle ElementType, U, p\rangle\rangle \}$, where p is the unique parameter.

- For each XML element p with a type U containing descriptors k_1 to k_n in d, map is $\{map(U) \otimes \langle\langle Value, k_1, p\rangle\rangle, ..., \langle\langle Value, k_n, p\rangle\rangle \}$. p is the unique parameter that identifies U. k_1 ... k_n is the set of n descriptors for rsv that are values of the element type U.

- Say R is an edge in d between two subdocuments A and B of d. Let E_1 and E_2 be element types, $\{ \langle\langle ElementType, E_1, p\rangle\rangle \} \in map(A)$ and $\{ \langle\langle ElementType, E_2, q\rangle\rangle \} \in map(B)$. We can then say that $map(R(AB)) = map(A) \otimes \{ \langle\langle R, p, q\rangle\rangle \} \otimes map(B)$. p and q are unique parameters. p is an identifier for E_1 and q for E_2.

In the GPX model, the XML documents are stored in the inverted index using as a key the location of each term identified by an absolute XPath expression [Geva, 2005]. This way the complete XML tree structure is preserved in the index. It is not stored in a dedicated index to be used in a post-processing step like in the XML vector space model. The translation is therefore the same as in the pure type XML retrieval model.

However, hierarchical inclusion $\trianglerighteq$ is not implemented, because Proposition 4.7.1 is not given: $rsv(\chi(d), \chi(q)) > 0$ does not mean $map(\chi(d)) \supseteq map(\chi(q))$. In the model, a leaf element containing 'house' and 'garden' is about a query asking for 'house'. But according to Proposition 4.7.1, $\{ \langle\langle ElementType, Section, i_1\rangle\rangle, \langle\langle Value, House, i_1\rangle\rangle, \langle\langle Value, Garden, i_1\rangle\rangle \}$ is not about $\{ \langle\langle ElementType, Section, i_1\rangle\rangle, \langle\langle Value, House, i_1\rangle\rangle \}$, as it has additional information about gardens. On account of the fact that we cannot use Proposition 4.7.1, we need to argue directly with the rsv function, as we have done for the XML vector space model.

Again, the Situation Theory definition can be misleading here, as the translation looks the same as for pure type XML retrieval. This is because the indexing creates a representation of the XML tree. However, the interpretation is completely different and closer to the Language Modelling II model with structured queries. Structure is not considered in itself but as a relationship between content in XML documents. This can be done, as there is the already in Section 4.7.1 explicated direct relationship between content components of an XML document and its corresponding XML tree.

The definitions of equivalence, composition and preclusion are the ones for pure type XML retrieval (using descriptor sets that contain all necessary information to calculate rsv) and can be omitted here. Containment will be different, because Proposition 4.7.1 does not hold. We need to include content in the containment relation and state a situation

S contains another situation T if T has only infons also found in S. This implies that containment is not just a condition of aboutness but leads directly to aboutness, as we see in the discussion of the reasoning rules next.

5.4.1.5 Rules

First we define the aboutness relation as a combination of rsv_L and rsv_R:

- Leaf elements L: $rsv_L = K^{n-1} \sum_{i=1}^{n} \frac{t_i}{f_i}$.
- Branch elements R: $rsv_R = D(c) \sum_{i=1}^{c} rsv_{L_i}$.

Let $\mathcal{X}$ be a set of XML documents, with q and $d \in \mathcal{X}$. Furthermore, let $\chi(d)$ and $\chi(q)$ be descriptor sets for XML elements identified by their XPath. The descriptors again include all information necessary to calculate rsv. The aboutness decision is then:

$$d \text{ about } q \text{ if and only if } rsv(\chi(d), \chi(q)) > 0$$

This is a significant difference from the pure type XML retrieval model for which we had a similar translation reflecting structure but also a structure-based aboutness definition using hierarchical inclusion. As explained, in the GPX model, structure is mainly considered as a relationship of content in an XML document. This leads us to this 'unstructured' aboutness definition. This combination of structured representation with 'unstructured' aboutness is a new creation for XML retrieval. Please also note, that this aboutness definition is equivalent to the flat model except that we additionally have to considere rsv_R. Next, we investigate the reasoning properties of the GPX model.

The model holds for **Reflexivity** and other reasoning rules that demonstrate how close it is to the pure type XML retrieval model. It does not support Symmetry or Transitivity. We prove Reflexivity first. We need to show that $S \Box\!\!\leadsto S$ ($S \equiv map(A)$) and assess whether we are talking about leaf or branch elements: (1) A leaf document component would be about itself as $t_i > 0$ and $f_i > 0$. (2) For branch elements, all leaf elements are about themselves. Then, also $rsv(A, A) > 0$, because $D(c) > 0$. Reflexivity holds.

The model preserves the structure of an XML tree in the index using XPath if the document is stored in the index according to the full XPath expression. According to [Geva, 2005], each term in an XML document is identified by three elements in the index: File path, absolute XPath context and term position within the XPath context. As a query language, however, GPX uses the INEX NEXI model [Geva, 2005], which is an (enhanced) subset of XPath. Because NEXI is only a subset of XPath, GPX, storing terms according to their full XPath expression, discriminates query and document representation without having a transformation function to map one onto the other.

According to [Geva, 2005], content-only queries are expressed as a search over the entire article element using NEXI. Therefore GPX, contrary to all the other XML retrieval models we have explored so far does not treat content-only queries differently from those using structural hints. Say, we have a query $//article[about(house)]$ that is answered by an element $article[1]/bm[1]/bib[1]/bibl[1]/bb[13]/pp[1]$. Then, we cannot swap them and use

$article[1]/bm[1]/bib[1]/bibl[1]/bb[13]/pp[1]$ as a query, as it is not a valid query expression. Thus, GPX does not support the **Symmetry** rule: If $S \,\Box\!\!\rightsquigarrow\, T$ then not $T \,\Box\!\!\rightsquigarrow\, S$. In some of the previous models, NEXI was also used in the query but terms in XML documents were identified not via an XPath expression but only with a single XML element like a section or an article. Query and document component both had the shape of sets of descriptors like $\{q_1, ..., q_n\}$ for queries, which could be substituted by $\{d_1, ..., d_2\}$ so that the document component would serve as the query.

Next, we look at **Transitivity**. As rsv is larger than 0 if $t_i > 0$ in Equation (5.4.1), the overlap of terms in query and document component determines aboutness. As shown in [Huibers, 1996], overlap aboutness decisions do not generally support Transitivity. Transitivity is part of the pure type XML retrieval and can be considered as part of those reasoning properties that indicate advanced XML structural reasoning, i.e. that aboutness is propagated from the leaves to the root of an XML document. Regarding pure type XML retrieval, it indicates that not only the immediate parent of an XML elements is taken into consideration but further ancestors, too.

It is interesting to see that GPX is not symmetric and also does not support Transitivity, although the latter is part of pure type XML retrieval reasoning. We ascribe this to the fact that for GPX XML structure is not considered in itself but as a relationship between content in XML documents. That is why Proposition 4.7.1 does not hold. Instead of XML structure directly the structure induced relationship between content informs the aboutness decision. This leads to a combination of reasoning properties one would expect from an aboutness decision fully incorporating hierarchical inclusion and one, which does not consider XML structure. Furthermore, GPX does not support Symmetry while LM II does, because in the latter model structure is only used to calculate the interpolation of parent and children language models. In GPX, however, structure is also used for the querying, and CO queries are taken to be a special case of CAS queries.

Set Equivalence is given, as the definition of equivalence means the subsituation of either complete branch elements or just leaf elements. Only the proof for Left Set Equivalence is presented. Say, that $S \equiv map(A)$, $T \equiv map(B)$ and $U \equiv map(C)$. Then, according to the assumptions $rsv(A, C) > 0$. If we exchange in Equation (5.4.1) all the t_i of A with the same set from B also $rsv(B, C) > 0$. Set Equivalence is given.

Left Monotonic Union (LMU) would be a property of the GPX aboutness systems, if with $S \,\Box\!\!\rightsquigarrow\, T$ we could derive that $S \otimes U \,\Box\!\!\rightsquigarrow\, T$. Let us assume that $S \equiv map(A)$, $T \equiv map(B)$ and $S \otimes U \equiv map(C)$. Thus, $rsv(A, B) > 0$. Three cases depending on c in Equation (5.4.2) have to be discriminated to consider the impact of $D(c)$: (1) For $c = 0$, we would clearly be able to then also say that also $rsv(A, C) > 0$. The sum in the calculation for leaf elements will at least stay the same when adding new information items. Sums in relevance calculation (as in Equation (5.4.1)) generally promote monotonic behaviour. Let us assume (2) $c = 1$. Several cases have to be considered. The interesting one is that the added information makes a neighbouring element about the information need. We therefore also boost the information contained in the parent elements with $D(c)$ increasing from 0.49 to 0.99, as c increases to 2. This can have a significant negative impact on the specificity value in content-only XML retrieval, as the following example shows.

Let us assume that we have a query asking for house and garden. We have one paragraph $P1$ containing exactly this information. This paragraph is part of a section $S1$ that is about 'house', 'garden' and 'garage'. This means that the section contains at least one more paragraph $P2$. The overall return is $P1$, as with $c = 1$ the decay factor will be $D(c) = 0.49$, which makes $S1$ less than half the value of $P1$. Now, we add with LMU another information about gardens to $P2$. $D(c)$ for $S1$ becomes 0.99. The score for $S1$ will be larger than the one for $P1$. It will become the best to return. The section will still give the most exhaustive information. Regarding specificity, however, the return of $S1$ instead of $P1$ means a loss in focus. This is particularly relevant for the INEX user model of finding the most focussed answers to content-only queries without overlap, as we shall see in Section 8.4, where we discuss the corresponding experimental behaviour. For GPX, this means that the system extracts from each path, leading from top-most element to leaf, the highest ranking element, which would be in this case $S1$, which is clearly less focussed (specific) than $P1$.

A possible improvement could be to penalise more branch elements for their number of relevant children. Otherwise, there could be a tendency towards rewarding exhaustivity. Another way would be to penalise the occurrence of non-relevant terms. That would definitely improve specificity. To summarise, the aboutness relation would not be changed for $c = 1$, as $rsv(A, C) > 0$, but the changes for specificity are possibly not desirable. For (3) $c > 1$, this will not occur, because $D(c)$ is 0.99 in any case. LMU is fully supported by GPX, as in all three cases $rsv(A, C) > 0$ whatever the impact on specificity for case 2.

Right Monotonic Union (RMU) on the other hand is not supported in GPX, which again shows how close it is to pure type XML retrieval. If RMU were supported, given that $D \Box\!\!\rightsquigarrow Q$ we could conclude that $D \Box\!\!\rightsquigarrow Q \otimes Q'$. As the model uses CAS expressions also for CO-queries, a typical query would look like $//X[about(//A, C)]$. We can merge this query with another one Q_1 $//Y[about(//A, Z)]$ to become Q_{new} $//X[about(//A, C)]$ $//Y[about(A, Z)]$.[1] Then, it is not necessarily the case that if $D \Box\!\!\rightsquigarrow Q$ then also $D \Box\!\!\rightsquigarrow Q_{new}$. RMU is not given. Cut and Right Weakening are both not supported, as they are not given for the flat document retrieval equivalent.

Mix holds because it is a special case of Left Monotonic Union. We can say that $S \otimes T \Box\!\!\rightsquigarrow U$, given $S \Box\!\!\rightsquigarrow U$ and $T \Box\!\!\rightsquigarrow U$. More interesting is **Context-Free And** at this point. Will it be supported, although Right Monotonic Union was not? Again, if Context-Free And is supported, this can be seen as an indicator of advanced structural reasoning in pure type XML retrieval. Are we allowed to assume that with $S \Box\!\!\rightsquigarrow T$ and $S \Box\!\!\rightsquigarrow U$, also $S \Box\!\!\rightsquigarrow T \otimes U$? Let us assume that $S \equiv map(A)$, $T \equiv map(B)$ and $U \equiv map(C)$. In addition to the assumptions for Right Monotonic Union, we know that $S \Box\!\!\rightsquigarrow U$ or $rsv(A, C) > 0$. According to our *map*, the compositions of two situations, which are about the same query, cannot lead to a situation, which is not about the query.

[1] In terms of the Situation Theory formalisation, we could add new element type and relational infons in order to query for a different substructure in the XML document.

Thus, with $rsv(A, C) > 0$, we know that both situations $T \otimes U$ together must also be answered by S. This means Context-Free And is supported.

Yet, we do not necessarily foster specificity, as the following example shows. Let us assume that we have a query about 'house' and 'garden' and a section also about 'house' and 'garden' as the fully focussed answer. Say, this section has a paragraph with 'house'. Now, we add to the query the further specification that we are looking for paragraphs. Then, the answer is the paragraph with 'house', although the more specific answer would have been the section. This is due to the fact that GPX XML retrieval is not based on hierarchical inclusion of query and document component, as acknowledged in the following quote [Geva, 2005]:

> 'Our [...] interpretation was to ignore the structural constraint altogether. This may not have been the wisest choice, and our CAS results were not quite as good as the CO/COS results.'

Surface containment holds. If $S_i \to T_i$ then also $S \,\Box\!\!\rightsquigarrow T$. Say $S \equiv map(A)$, $S_i \equiv map(A')$, $T \equiv map(B)$ and $T_i \equiv map(B')$. Then according to the definition of $\to$, also $rsv(A', B') > 0$. This implies that $rsv(A, B) > 0$, because we have for the t_i in Equation (5.4.1) common to A and B at least the ones in both A' and B', which proves Surface Containment. Please note the distinct difference to pure type XML retrieval for which Containment was rather a condition of aboutness, as demonstrated in Section 4.7.4.

Looking at **Absorption** next, with $S \to T$ given, we can conclude that $S \otimes T \equiv S$ because of the definitions of composition and containment. **Right Containment Monotonicity** is not given, as RMU is not supported. **Non-conflict-containment** and **Containment Preclusion** are obviously given because of the general definition of containment.

Mutual Preclusion is given. **Simple Anti-Aboutness** is a condition we would like to exclude for this model. **Negation Rational** and **Strict Negation Rational** do not hold. We demonstrate only the proof of Negation Rational, where a $\{\langle\langle ElementType, Section, i_1; 1\rangle\rangle, \langle\langle Parent, i_1, i_2; 1\rangle\rangle, \langle\langle ElementType, Paragraph, i_2; 1\rangle\rangle, \langle\langle Value, house, i_2; 1\rangle\rangle\}$ is not about a query $\{\langle\langle ElementType, Paragraph, i_2; 1\rangle\rangle, \langle\langle Value, courtyard, i_2; 1\rangle\rangle\}$, but is about a query $\{\langle\langle ElementType, Paragraph, i_2; 1\rangle\rangle, \langle\langle Value, courtyard, i_2; 1\rangle\rangle, \langle\langle Value, house, i_2; 1\rangle\rangle\}$. For readability reasons, we ignore term frequency and document frequency and just use index terms. We also do not assume the **Closed World Assumption**. There can be other information than the one in S_i and T_j that would make S to be about T.

Guarded and **Qualified Left Monotonicity** hold because Left Monotonicity does. **Guarded** and **Qualified Right Monotonicity** are not be supported, as Right Monotonic Union is not.

5.4.1.6 Completeness

We have to show that for two XML documents A, B: If $rsv(A, B) > 0$ then $map(A) \,\Box\!\!\rightsquigarrow map(B)$. We do that for branch and leaf elements. Let us assume $rsv(A, B) > 0$ and that $S \equiv map(A)$ and $T \equiv map(B)$. We assume that U is the subsituation of S and T that leads to $rsv(A, B) > 0$, i.e. it contains all infons of t_i common to A and B. And, W are all the other subsituations in S except for U. For leaf elements L: With Reflexivity $U \Box\!\!\rightsquigarrow U$. Then, with Left Monotonic Union ($W \otimes U \Box\!\!\rightsquigarrow U$). Furthermore, with Left Set Equivalence: $S \Box\!\!\rightsquigarrow U$. According to the assumption $rsv(A, B) > 0$: If the situation T is added to an existing situation, which S is about, this aboutness relation is not changed. Thus, we can apply Context-Free And: $S \Box\!\!\rightsquigarrow U \otimes T$. With the definition of subsituations and Absorption: $S \Box\!\!\rightsquigarrow T$. For branch elements R, the proof is the same as for leaves.

5.4.1.7 Reflection

The reflection properties are the same as for the language models from Section 5.3, but their justification differs:

1. The bottom exhaustive document component is $\{\emptyset\}$. $\{\emptyset\}$ is never exhaustively about any query, as only in this case $\sum_{i=1}^{n} \frac{t_i}{f_i}$ in Equation (5.4.1) is guaranteed to be always 0.

2. The bottom exhaustive query is never exhaustively answered by any document component and is also $\{\emptyset\}$. The proof is the same as for the bottom exhaustive document component.

3. The bottom specific document component is never specifically about any query and is $\{\emptyset\}$. The proof is analogous to the one for bottom exhaustive document components.

4. The bottom specific query is again $\{\emptyset\}$.

This reflection clearly shows that content relations dominate the model. The difference in specificity and exhaustivity reasoning stems directly from the way the query is formalised compared to the document representation. In the rest of reasoning, however, specificity and exhaustivity cannot be differentiated. A better specificity is not the result of identifying the best focus in the content but of the filters in the NEXI queries.

5.4.1.8 Conclusion

Table 5.29 summarises the results of our theoretical evaluation for GPX. Comparing it to Tables 5.10 and 5.25 and therefore XML vector space and XML language model retrieval, we can clearly see how comparably close GPX is to pure type XML retrieval. As seen in Table 5.25, both language modelling approaches, we have analysed, are largely identical to their flat model equivalent. XML vector space retrieval exhibits more reasoning similarities with pure type XML retrieval but is still symmetric. Though GPX is not supporting Transitivity and therefore a key characteristics of XML-related reasoning behaviour, it is not symmetric and thus able to distinguish exhaustivity ($D \Box\!\!\rightsquigarrow Q$) from specificity ($Q \Box\!\!\rightsquigarrow D$).

According to Table 5.10, XML vector space retrieval uses thresholds to adjust its monotonic reasoning behaviour to the requirements of XML retrieval. GPX does not need

Reasoning behaviour	*Plain GPX*	*XML GPX*	*Pure Type XML Retrieval*
Singleton Reflexivity	fully	N/A	N/A
Reflexivity	N/A	fully	fully
Symmetry	fully	not	not
Set Equivalence	fully	fully	fully
Transitivity	not	not	fully
Euclid	not	not	not
LMU	fully	fully	fully
RMU	fully	not	not
Cut	not	not	fully
Right Weakening	not	not	not
Mix	fully	fully	fully
Context-Free And	fully	fully	fully
Containment	fully	fully	not
Containment Composition	fully	fully	fully
Absorption	fully	fully	not
Right Containment Monotonicity	fully	not	not
Non-Conflict-Containment	N/A	fully	fully
Containment Preclusion	N/A	fully	fully
Mutual Preclusion	N/A	fully	fully
Negation Rational	N/A	not	fully
Closed World Assumption	N/A	not	not

Table 5.29: GPX retrieval evaluation results

to adjust its behaviour, as it has been specifically designed for XML retrieval. In fact, in its monotonic reasoning abilities it shows almost identical behaviour to pure type XML retrieval. Cut reasoning aside, both have the same behaviour for both monotonic unions as well as Mix and Context-Free And. As we show in Chapter 8, this is one reason for its convincing behaviour in the experimental evaluation. However, it does not always experimentally outperform XML vector space retrieval, which shows that having the same reasoning behaviour as pure type XML retrieval does not necessarily mean that a model is better than other models, as we have already discussed in Section 4.7. That depends very much on other factors, too. For instance, such factors are the experimental evaluation task or the way the content of XML elements plays a role in the aboutness decision of a model. In this case, we have seen that for GPX structure is a relationship of content in XML documents in most of its reasoning. Table 5.29 shows that Containment, as a reasoning property, that very much depends on content relationships, is supported by GPX, while it is not supported by pure type XML retrieval and XML vector space retrieval.

In conclusion, it is this combination of a representation of structure in the translation with an aboutness decision that neglects structure, that makes GPX so interesting. Again, we can see the typical approach in XML retrieval to combine the evidence from XML structure with content relationships known from flat document retrieval.

Next we analyse another model called contextualisation also specially designed for XML retrieval.

5.4.2 Contextualisation Model

5.4.2.1 Background

[Arvola et al., 2005a] and [Arvola et al., 2005b] introduce a new re-weighting method called *Contextualization*. In this model, the ancestors of an element are considered to be its context. The approach takes into account any level of hierarchy of ancestors and differs therefore from any other XML retrieval model we have met so far. For the model, the parent of an element is its first level context, the grandparent its second level context and so on. Those elements in a strong context are rewarded by being higher ranked, while those in a worse context are penalised by being lower ranked.

In [Arvola et al., 2005a], the weighting scheme is based on the probabilistic retrieval framework BM25 [Robertson et al., 1992]:

$$w(k,\xi) = \frac{kf_\xi}{kf_\xi + v \times ((1-b) + b\frac{\xi f_c}{\xi f_k})} \times \frac{log(\frac{N}{m})}{log(N)} \tag{5.4.3}$$

kf_ξ is the number of times term k can be found in element ξ. The model indexes all content elements and stores where they can be found in the XML document structure. N is the total number of content elements in the collection, m the number of content elements with k. ξf_c is the number of all descendant content elements of ξ, while ξf_k is the number of descendant content elements of the ξ containing key k. v and b are tuning constants.

A query term qt can be prefixed with $+$ or $-$ to increase or decrease its importance:

$$\begin{aligned} w(+qt,\xi) &= +w(qt,\xi) \qquad (5.4.4) \\ w(-qt,\xi) &= -w(qt,\xi) \end{aligned}$$

Using these prefixes for query terms, the model implements a more conservative approach to monotonicity, as we will see in Section 5.4.2.5. Overall the ranking is calculated for a query q by averaging the query terms:

$$w(q,\xi) = \frac{\sum_{i=1}^{n} w(qt_i,\xi)}{n} \tag{5.4.5}$$

Next to the basic weighting scheme, the authors employ their contextualisation method to adjust basic retrieval to the needs of XML retrieval. Using this method, elements are re-ranked based on the weights of their ancestors. [Arvola et al., 2005b] use four contextualisation functions, based on the their experiences in [Arvola et al., 2005a], where they developed a general contextualisation function C:

$$C(q,\xi,g) = \begin{cases} 0 & \text{if } w(q,\xi) = 0 \\ \frac{\sum_{i=1}^{len(\xi)} g[i] \times w(q,\delta_i(\xi))}{\sum_{i=1}^{len(\xi)} g[i]} & \text{, otherwise} \end{cases} \tag{5.4.6}$$

w is a weighting function. g is called contextualisation vector, represented by a tuple, consisting of values by which elements between the root element and the ξ element are weighted. Contextualisation is applied only to those elements whose basic weight is not 0. The contextualised weights of elements are calculated by weighted average.

The first contexualisation method is called *root* contexualisation c_r [Arvola et al., 2005b]:

$$c_r(q,\xi) = \frac{w(k,\xi) + 1.5 * w(q,\delta_1(\xi))}{2.5} \tag{5.4.7}$$

The second contexualisation method is called *parent* contexualisation c_p. It is an average of the weights of an element and its parent:

$$c_p(q,\xi) = \frac{w(k,\xi) + w(q,\delta_{len(\xi)-1}(\xi))}{2} \tag{5.4.8}$$

The third contextualisation is the tower contextualisation and an average of the weights of an element and all its ancestors:

$$c_t(q,\xi) = \frac{\sum_{i=1}^{len(\xi)} w(q,\delta_i(\xi))}{len(\xi)} \tag{5.4.9}$$

The forth contexualisation is called root and tower contextualizaton and is a combination of the two:

$$c_t(q,\xi) = \frac{\sum_{i=1}^{w(q,\delta_1(\xi))+len(\xi)} w(q,\delta_i(\xi))}{len(\xi)+1} \tag{5.4.10}$$

Next, we look at our example calculation.

5.4.2.2 Example

Table 5.30: kf_ξ

	D1	D2	D3	P_{11}	P_{12}	P_{21}	P_{22}	P_{31}	P_{32}
house	1	0	1	1	0	0	0	1	0
garden	0	2	0	0	0	1	1	0	0
courtyard	0	1	1	0	0	0	1	0	1
garage	1	0	1	1	0	0	0	0	1
damage	1	0	0	0	1	0	0	0	0
fire	1	0	0	0	1	1	0	0	0
door	0	1	0	0	0	0	1	0	1
arrive	0	1	1	0	0	0	0	0	1

Table 5.31: ξf_k

	D1	D2	D3	P_{11}	P_{12}	P_{21}	P_{22}	P_{31}	P_{32}
house	1	0	1	0	0	0	0	0	0
garden	0	2	0	0	0	0	0	0	0
courtyard	0	1	1	0	0	0	0	0	0
garage	1	0	1	0	0	0	0	0	0
damage	1	0	0	0	0	0	0	0	0
fire	1	0	0	0	0	0	0	0	0
door	0	1	0	0	0	0	0	0	0
arrive	0	1	1	0	0	0	0	0	0

We employ the same example as for all the other models (Section 5.2.2). We choose $v = 2$,

Table 5.32: m

house	garden	courtyard	garage	damage	fire	door	arrive
2	2	2	2	1	1	1	2

$N = 6$ and $b = 0.1$, as they have been chosen in the experiments in [Arvola et al., 2005b]. Then, in the first step kf_ξ is calculated as in Table 5.30.

Table 5.33: BM25 Results

	D1	D2	D3	P_{11}	P_{12}	P_{21}	P_{22}	P_{31}	P_{32}
house	0.198	0	0.198	0.219	0	0	0	0.219	0
garden	0	0.307	0	0	0	0.219	0.219	0	0
courtyard	0	0.198	0.198	0	0	0	0.219	0	0.219
garage	0.198	0	0.198	0.219	0	0	0	0	0.219
damage	0.313	0	0	0	0.357	0	0	0	0
fire	0.313	0	0	0	0.357	0	0	0	0
door	0	0.313	0	0	0	0.357	0	0	0
arrive	0	0.198	0.198	0	0	0	0.219	0	0.219

Table 5.34: Tower Contexualisation

	D1	D2	D3	P_{11}	P_{12}	P_{21}	P_{22}	P_{31}	P_{32}
house	0.198	0	0.198	0.209	0	0	0	0.209	0
garden	0	0.307	0	0	0	0.263	0.263	0	0
courtyard	0	0.198	0.198	0	0	0	0.209	0	0.209
garage	0.198	0	0.198	0.209	0	0	0	0	0.209
damage	0.313	0	0	0	0.335	0	0	0	0
fire	0.313	0	0	0	0.335	0	0	0	0
door	0	0.313	0	0	0	0.335	0	0	0
arrive	0	0.198	0.198	0	0	0	0.209	0	0.209

ξf_c for $D1$, $D2$ and $D3$ is 2, while for the other elements it is 0. ξf_k is given in Table 5.31 and m in Table 5.32. The overall results after calculating Equation (5.4.3) are in Table 5.33. The ranking is then $D2(0.168)$, $P_{22}(0.146)$, $D3(0.132)$, $P_{21}(0.073)$, $P_{31}(0.073)$, $P_{32}(0.073)$, $P_{11}(0.073)$, $D1(0.066)$ and $P_{12}(0)$.

As an example for contextualisation let us apply tower contextualisation, which for our limited example is equivalent to parent contextualisation. The results in Table 5.34 show a re-weighting of the children elements dependent on the strength of their context (the weight of their parents).

After applying tower contextualisation, the ranking is $D2(0.168)$, $P_{22}(0.157)$, $D3(0.132)$, $P_{21}(0.07)$, $P_{31}(0.07)$, $P_{32}(0.07)$, $P_{11}(0.07)$, $D1(0.066)$ and $P_{12}(0)$. One can clearly see that P_{22} is now stronger emphazized, which reflects the fact that it is in the strong context of its parent $D2$.

5.4.2.3 Flat Model Equivalent

Following our methodology, let us first investigate the flat model equivalent, which equates to a document level discussion of the underlying BM25 model. Let $\mathcal{D}$ be a set of documents and d be a document in it, while q is a query. $\chi(d)$ and $\chi(q)$ are descriptor sets with keys and all the other information necessary to calculate the weight of a key in a document component. Then:

$$d \text{ about } q \text{ if and only if } rsv(\chi(d), \chi(q)) > 0$$

$rsv(\chi(d), \chi(q)) = \frac{\sum_{i=1}^{n} w(qt_i, \xi)}{n}$, where $w(k, \xi) = \frac{kf_\xi}{kf_\xi + v \times ((1-b) + b\frac{\xi fc}{\xi f_k})} \times \frac{log(\frac{N}{m})}{log(N)}$. The basic infon language describes the transformation of keys into situations:

$$map(\chi(d)) = \{\, \langle\langle Value, t; 1\rangle\rangle \,|t \in \chi(d)\}$$

Equivalence, composition, containment and preclusion are similarly defined as in the flat model equivalent for GPX from Section 5.4.1.3, because we use the same basic infon language and both models are based on information overlap.

Reflexivity holds. Let us assume that $map(A) \equiv \{\phi\}$ and $map(B) \equiv \{\phi\}$. A and B are descriptor sets. Then, $rsv(A, B) > 0$ only if $N \neq m$, as otherwise $log(\frac{N}{m}) = 0$ and $rsv(A, B) = 0$ in Equation (5.4.3). We assume that the collection contains more elements than the one that has ϕ and Reflexivity holds.

With regard to **Symmetry**, we can also conclude from the assumption $S \Box\!\!\leadsto T$ that $T \Box\!\!\leadsto S$. $S \equiv map(A)$ and $T \equiv map(B)$. kf_ξ describes an information overlap, the number of times a key can be found in document and query. It will be > 0 whether we conclude $S \Box\!\!\leadsto T$ or $T \Box\!\!\leadsto S$. Thus, Symmetry is given.

Transitivity is not given. The key k could be found in two indexes without being found in a third. We could easily construct a counter-example to show that $S \Box\!\!\leadsto T$ and $T \Box\!\!\leadsto U$, but $S \Box\!\!\not\leadsto U$. **Euclid** is not supported either for similar reasons as in other overlap-based models. **Set Equivalence** is supported for both Left and Right Set Equivalence. We can substitute kf_ξ using an equivalent set of keys without changing the aboutness relation.

Regarding the combination rules, **Left Monotonic Union** (LMU) is conditionally supported. With $S \Box\!\!\leadsto T$, we would be able to say $S \otimes U \Box\!\!\leadsto T$. Let us assume that $S \equiv map(A)$, $T \equiv map(B)$ and $S \otimes U \equiv map(C)$. Looking at the overall ranking function $w(q, \xi)$ in Equation (5.4.4), it only becomes 0 if the $\sum$ becomes 0. According to the assumptions therefore $rsv(A, B) > 0$. However, if the U is the document situation for unwanted query terms then it could be the case that $rsv(A, C) \leq 0$ according to Equation (5.4.4). Thus, LMU holds only under the condition that the newly added information in the document components does not negatively outweigh the existing information.

For **Right Monotonic Union**, we have the mirror case. This time, we extend the query situation so that with $D \Box\!\!\leadsto Q$ also $D \Box\!\!\leadsto Q \otimes Q'$. Say, $D \equiv map(A)$, $Q \equiv map(B)$ and $Q \otimes Q' \equiv map(C)$. In case this new query situation is prefixed by '-', it could be the case that $rsv(A, C) \leq 0$. Therefore, Right Monotonic Union is only conditionally supported.

If **Cut** were given, then $D \Box\!\!\leadsto Q$, with $D \otimes D' \Box\!\!\leadsto Q$ and $D \Box\!\!\leadsto D'$. Cut does not hold, n could be become 0 in Equation (5.4.4) if all the relevant information were in D'. **Right Weakening** is also not supported. We could change the query size so that $n = 0$. **Mix** and **Context-Free And** are supported, because Left Monotonic Union and Right Monotonic Union both hold. They are unconditionally supported as the added information has to be relevant for both.

Deep containment is not supported but only **surface containment**. As for the flat GPX model, a subsituation would be a subset of the keywords forming the original situation. **Containment Composition** is then obviously supported as is **Absorption**. **Right Containment Monotonicity** holds only conditionally as a special case of RMU. All the non-aboutness rules are not applicable. The **Closed World Assumption** does not apply either.

Next, we study the full Contextualisation model.

5.4.2.4 Translation

The translation and the definition of the operators such as equivalence, etc. are analogous to XML vector spaces and can be omitted here.

Please note that we could assume that this translation should more look like the one for pure types, as both models index the structure of XML documents. Therefore we might think that we can again use the pure type *map* function, as we did for GPX. In the model, however, this hierarchical nature is not used to constrain the retrieval of content structurally, as it has been the case in the GPX model. Only content elements are indexed and document components are simply identified by an XPath. The XML relationships between elements, however, are not used in the Equation (5.4.3). They are only used in the contextualisation step to interpolate elements with their ancestors. As we shall see next, this has little to no impact on the reasoning behaviour. The model is therefore comparable to XML vector spaces in that it indexes only particular elements. In this case only the content elements.

5.4.2.5 Rules

Let d and q be a document component and a query, and let $\chi(d)$ and $\chi(q)$ be their descriptor sets with all the information necessary for rsv. The contextualisation aboutness decision is:

$$d \text{ about } q \text{ if and if only } rsv(\chi(d), \chi(q)) > 0$$

rsv is defined as in the flat equivalent, only that now the Contextualisation $C(x)$ is added: $rsv(\chi(d), \chi(q)) = \frac{\sum_{i=1}^{n} w_c(qt_i, \xi)}{n}$ and $w_c(k, \xi) = C(\frac{kf_\xi}{kf_\xi + v \times ((1-b) + b\frac{\xi f_c}{\xi f_k})} \times \frac{log(\frac{N}{m})}{log(N)})$. w_c is the contextualized weight.

We do not have to show that **Singleton Reflexivity** is still given, as the model does not change the fundamentals of its flat equivalent. The contextualisation is an example of re-weighting relevant and non-relevant elements, as we have seen with respect to P_{22} in the example calculation from Section 5.4.2.2. It does not change the underlying aboutness relations. Therefore again $\{\phi\} \,\Box\!\!\rightsquigarrow \{\phi\}$. This is true for all contextualisations, because a singleton document component cannot have a relevant context.

Symmetry is given, for the same reason as it is supported in the flat equivalent of the model. The contextualisation step has no impact here. **Transitivity** does not hold, as it is not given for the flat equivalent either.

Set Equivalence is given. Let us assume that $S \equiv map(A)$, $T \equiv map(B)$ and $U \equiv map(C)$. A, B and C are descriptor sets. Then, $A \equiv B$ and $rsv(A, C) > 0$. The element will not change if we substitute all information in A with the equivalent ones from B. Thus, $rsv(B, C) > 0$.

Set Equivalence holds. However, considering the contextualisation function, the context could play a decisive role to alter the behaviour. Just because $S \equiv T$ is given, it does not mean that the context of these two situations is still the same. Therefore, Set Equivalence does not mean the same relevance result though aboutness might be given. We can already see here that it might have been interesting to look at a different aboutness decision that would have included a threshold θ for rsv. Then, Set Equivalence aboutness could be changed through a different context and the contextualisation function. **Euclid** does not hold, as it was not given for the flat model equivalent.

If **Left Monotonic Union** held, we could with $S \square\!\!\rightsquigarrow T$ conclude that $S \otimes U \square\!\!\rightsquigarrow T$. Let us assume that $S \equiv map(A)$, $T \equiv map(B)$ and $S \otimes U \equiv map(C)$. It is again conditionally supported, as the addition of a strongly not desired information could reduce $rsv(C, B) \leq 0$. The contextualisation has further impact onto this. Any of the above contextualisation functions could lead to a new context, in which rsv would become 0 or less than 0. A document component that is about a query can be placed in a context of other document components with highly unwanted document components. Regarding the parent contextualisation, this would be the parent component, which would mean that some of the siblings are about undesired query items. For the root contextualisation, the overall article could contain too much undesired information. Regarding tower contextualisation, the relatives of the elements could contribute too much undesired information.

Right Monotonic Union is again the mirror case of LMU. With $S \square\!\!\rightsquigarrow T$, we can conditionally say that $S \square\!\!\rightsquigarrow T \otimes U$. **Cut** is not supported for the XML retrieval Contextualisation model, as it was not supported for the flat equivalent. Looking at the impact of contextualisations, Cut — even if not directly leading to non-aboutness — could leave us with a highly undesirable context, therefore negating an existing aboutness. **Right Weakening** is not supported for similar reasons as why Cut is not given. It could also have a decisive impact on the contextualisation.

If **Mix** held, we would be able to conclude $S \otimes T \square\!\!\rightsquigarrow U$, given that $S \square\!\!\rightsquigarrow U$ and $T \square\!\!\rightsquigarrow U$. Say, $S \equiv map(A)$, $T \equiv map(B)$ and $U \equiv map(C)$. Mix is a special case of Left Monotonic Union. Thus, it is supported at least conditionally. Left Monotonic Union was only conditionally supported, as the newly added information could have been either undesirable information or undesirable context. Both cannot be the case for Mix, because the added situation is also about U. Thus, Mix is unconditionally given. The situation is similar for **Context-Free And**, a special case of Right Monotonic Union. We can say that with $S \square\!\!\rightsquigarrow T$ and $S \square\!\!\rightsquigarrow U$ also $S \square\!\!\rightsquigarrow T \otimes U$.

Surface containment holds for the same reasons as for the flat model equivalent.

With $S_i \rightarrow T_i$, we also have $S \Box\!\!\rightsquigarrow T$. **Absorption** is supported for similar reasons as for GPX. **Right Containment Monotonicity** would conclude $S \Box\!\!\rightsquigarrow U$, given that $S \Box\!\!\rightsquigarrow T$ and $T \rightarrow U$. Say, $S \equiv map(A)$, $T \equiv map(B)$ and $U \equiv map(C)$. Right Containment Monotonicity is only conditionally given. Again we must not add undesired information. In this case, situation T could contain situation U and $S \Box\!\!\rightsquigarrow T$, but still too much undesirable information could be in U to conclude $S \Box\!\!\rightsquigarrow U$. Right Containment Monotonicity is only conditionally given.

Non-conflict-containment and **Containment Preclusion** are obviously given because of the general definition of containment. **Mutual Preclusion** is not given, because of the definition of $\otimes$ and preclusion. For similar reasons, **Negation Rational** and **Strict Negation Rational** also do not hold. We do not repeat the proofs here, as they are the same as the one form Section 5.4.1.5 for GPX, which has the same translation and definition of preclusion. The model does not support the **Closed World Assumption**. There can be other information than the one in S_i and T_j that would make S to be about T.

Guarded and **Qualified Left Monotonicity** hold conditionally, because Left Monotonicity does. Furthermore, **Guarded** and **Qualified Right Monotonicity** are conditionally supported, as Right Monotonic Union is. The interesting point here is that the guards and qualification in these monotonic reasoning rules are related to preclusion and have therefore no impact on the conditions of LMU and RMU, which are based on negative weights.

5.4.2.6 Completeness

The completeness proof is similar to the one for XML vector spaces from Section 5.2.5 and can be omitted here.

5.4.2.7 Reflection

The reflection is similar to GPX from Section 5.4.1.7 with different justifications. Bottom exhaustive query and document component as well as the bottom specific query and document component are all $\{\emptyset\}$.

5.4.2.8 Conclusion

Table 5.35 summarises the results of our theoretical evaluation of the Contextualisation method. The Table looks very similar to Table 5.10 for XML vector space retrieval. Both Contextualisation and vector space retrieval support Symmetry and Transitivity. They differ from the pure type reasoning not just in these rules but also in many other. Contextualisation, too, uses conditions to adjust its monotonic reasoning behaviour for LMU and RMU as well as Mix and Context-Free And. They are also external conditions but this time they are even more external to the aboutness reasoning than the XML vector space retrieval ones, as they mostly depend on direct intervention by users stating which information they do not want. The sum of the relevance weights for negative queries terms must not be larger than the one for positive query terms.

Reasoning behaviour	*BM25*	*Contextualisation Method*	*Pure Type XML Retrieval*
Singleton Reflexivity	fully	fully	N/A
Reflexivity	fully	fully	fully
Symmetry	fully	fully	not
Set Equivalence	fully	fully	fully
Transitivity	not	not	fully
Euclid	not	not	not
LMU	$\sum -w < \sum +w$	$\sum -w < \sum +w$	fully
RMU	$\sum -w < \sum +w$	$\sum -w < \sum +w$	not
Cut	not	not	fully
Right Weakening	not	not	not
Mix	fully	fully	fully
Context-Free And	fully	fully	fully
Containment	fully	fully	not
Absorption	fully	fully	not
Right Containment Monotonicity	$\sum -w < \sum +w$	$\sum -w < \sum +w$	not
Non-Conflict-Containment	N/A	fully	fully
Containment Preclusion	N/A	fully	fully
Mutual Preclusion	N/A	fully	fully
Negation Rational	N/A	not	fully
Closed World Assumption	N/A	not	fully

Table 5.35: Contextualisation retrieval evaluation results

Looking back at Equation (5.4.3), one more time, a better threshold seems to be a good option for the model to further increase performance:

$$w(k,\xi) = \frac{kf_\xi}{kf_\xi + v \times ((1-b) + b\frac{\xi f_c}{\xi f_k})} \times \frac{log(\frac{N}{m})}{log(N)}$$

We could introduce a threshold similar to the one for XML vector space retrieval. We could introduce this threshold in Equation (5.4.6) and state that $C(q,\xi,g) = 0$ if $w(q,\xi) < \theta$. Then, we could use some more of the reasoning in the model. Looking at the first part of Equation (5.4.3), it is clear the fraction is closer to 1 the smaller $v \times ((1-b) + b\frac{\xi f_c}{\xi f_k})$ is. The size of $v \times ((1-b) + b\frac{\xi f_c}{\xi f_k})$ depends on the tuning parameters b and v, but also on $\frac{\xi f_c}{\xi f_k}$ or whether many descendants of ξ also contain k. This is an interesting statement, as it implies that an element ξ is more relevant if it has many relevant children. That is why $D2$ is strongly emphasized as the most relevant element in the ranking of our example calculation from Section 5.4.2.2. However, in Equation (5.4.6), the reasoning that elements with many relevant children are themselves even more relevant is lost, because there is no threshold deciding whether an element is relevant enough to contribute to the aboutness decision. As long as it is somewhat relevant, an element with less relevant children will be as much about a query as an element with many relevant children. As the contextualisation functions re-weigh the importance of elements, the contextualisation could also have more impact on the aboutness decision, if a threshold like the XML vector space one is introduced. In the current aboutness decision this is not the case, as Contextualisation does not decide on an element being relevant or not but only on the degree of relevance. This degree is currently not included in the aboutness decision, while in XML vector space retrieval it is.

Let us just briefly discuss, how, for instance, Left Monotonic Union reasoning would be influenced by such a change. For LMU, we also know that $S \otimes U \Box\!\!\rightsquigarrow T$ if $S \Box\!\!\rightsquigarrow T$. Furthermore, let us assume we have two highly relevant elements, where one is the parent $(S \otimes U)$ of the other (S). Yet, this parent also has many irrelevant children in U. With a thresholded aboutness decision, this means that $\frac{\xi f_c}{\xi f_k}$ can become so large that the overall threshold θ might be missed. Using θ, we can therefore differentiate the aboutness of a highly relevant and focussed element from the non-aboutness of its also highly relevant but non-focussed parent element. This is the kind of reasoning XML retrieval systems should support.

Contextualisation and XML vector space retrieval support Symmetry, Transitivity and other properties, which make them different from pure type XML retrieval. Contextualisation, just like XML vector space retrieval, uses external conditions to adjust its monotonic reasoning behaviour. Yet, contrary to XML vector space retrieval, these conditions depend on intervention by users, stating which information they do not want. Contextualisation is the only INEX model we investigate that has tried to make use of the ability to assign negative weights to query terms. All the others have decided to ignore negative weights for the query. There are further detailed differences between the reasoning behaviour of XML vector space retrieval and Contextualisation, which help explain the worse experimental performance of the latter (see Chapter 8). One example is that Contextualisation supports Containment, while XML vector space retrieval does not.

5.5 Conclusion

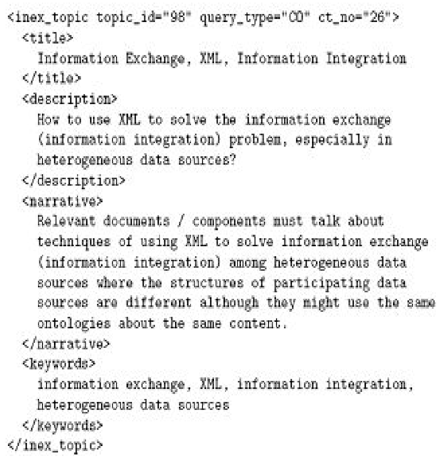
This chapter has applied our methodology to theoretically compare XML retrieval behaviour to five strong models from INEX. We have been able to show commonalities as well as differences between those models. The main commonality is that none of the presented models radically breaks with methods applied in flat document retrieval. XML structure is never directly included in the aboutness decision. The main difference in the models then is how they attempt to adjust a flat document retrieval model to the specific requirement of XML retrieval to deliver focussed answers. Here, the control of monotonic reasoning behaviours and other standard reasoning rules like Symmetry and Transitivity, have been found to be particularly important.

In our theoretical evaluation of five XML retrieval models, we could see how XML retrieval work is concentrated on the control of monotonic behaviour and other reasoning like Symmetry that heavily influence the primary aim of XML retrieval aboutness decisions, which is to find the most focussed answer. The importance of making the reasoning conservatively monotonic can be found in many retrieval strategies using internal and external thresholds. Thresholds have been a successful strategy to adjust the behaviour of flat document retrieval models towards the requirements of XML retrieval, as we have seen, e.g., in Section 5.2.

We finally assess in Chapter 8 how the reasoning behaviour of XML retrieval models leads to particular experimental performance at INEX. Yet, before we can discuss this experimental performance we first need to analyse the important XML retrieval method of filtering in Chapter 7, as it is used to support the delivery of only the most focussed elements in the experimental evaluation. Furthermore, we need to understand more about the underlying reasoning principles of the experimental evaluation when we aim to theoretically evaluate experimental evaluation in XML retrieval. To this end, we turn to Chapter 6.

Chapter 6

Theoretical Evaluation of the INEX Experimental Evaluation Methodology



```
<inex_topic topic_id="98" query_type="CO" ct_no="26">
  <title>
    Information Exchange, XML, Information Integration
  </title>
  <description>
    How to use XML to solve the information exchange
    (information integration) problem, especially in
    heterogeneous data sources?
  </description>
  <narrative>
    Relevant documents / components must talk about
    techniques of using XML to solve information exchange
    (information integration) among heterogeneous data
    sources where the structures of participating data
    sources are different although they might use the same
    ontologies about the same content.
  </narrative>
  <keywords>
    information exchange, XML, information integration,
    heterogeneous data sources
  </keywords>
</inex_topic>
```

Figure 6.1: Content-only topic in INEX 2003

6.1 Introduction

In this chapter, we explore a new domain for a theoretical evaluation. We investigate how to evaluate existing experimental evaluations. To my knowledge, this has not been done before. We proceed as follows: In Section 6.2 we briefly recall the basics of the INEX test collections while in Section 6.3, we discuss in more detail the relationship of the INEX test collections to the evaluation scales. In Section 6.4, we introduce models for reasoning that comply with these evaluation scales, before we bring all the introduced concepts together in the actual theoretical evaluation of the experimental evaluation in INEX 2004 and 2005 in Section 6.5.

6.2 INEX Test Collections

As already briefly discussed in Section 2.3, INEX created a test collection consisting of predefined query topics, a document collection and relevance assessments [Kazai and Lalmas, 2005]. The INEX 2005 collection uses the full texts of more than 10,000 IEEE articles — all marked up in XML. 12 magazines and 6 transactions are collected — from 1995 to 2002. The collection of INEX document components has a total size of 494 megabytes in size. The articles have varying length, with an average of 1,532 XML components and an average component depth of 6.9 [Kazai and Lalmas, 2005]. All in all, eight millions document components come together from table entries to whole articles. From INEX 2006 onwards, the much larger wikipedia collection has been used.

As discussed in Section 2.3, the INEX query language NEXI allows for the specification of structural query conditions, and INEX has defined two types of topics to reflect this. Content-only (CO) queries are standard IR retrieval tasks similar to those used in TREC. Content and structure (CAS) queries use both structure and content for formulating an information request. The structure might refer to the content of specific elements. An example would be a request demanding a paragraph about an information need. Furthermore, the query might ask for a certain element type like sections that are supposed to be retrieved. As in TREC, an INEX topic consists of the standard title, description and narrative fields. Figure 6.1 shows an example of such a CO query from [Kazai and Lalmas, July 2005]. One can see the similarity to standard TREC query types.

Next we introduce the INEX evaluation scales based on the evaluation dimensions of exhaustivity and specificity.

6.3 INEX Evaluation Scales

As seen in Chapter 2, for INEX the aim of XML retrieval is to retrieve not only relevant document components, but those at the right level of granularity, i.e. those that specifically answer a query. To evaluate how effective XML retrieval approaches are, it is necessary to consider whether the 'right' level is correctly identified. For this purpose, two evaluation criteria have been the basis for INEX to consider the structure when evaluating XML retrieval effectiveness, which we now want to look at in more detail.

As seen in Section 2.3, INEX has two evaluation dimensions:

- Topical exhaustivity reflects the extent to which the information contained in a document component satisfies the information need.
- Component specificity reflects the extent to which a document component focuses on the information need.

Specificity and exhaustivity are first used in IR literature to describe properties of the set of indexing terms assigned to a document [Kazai and Lalmas, July 2005]. INEX uses them more in an aboutness sense to name properties of document components. The history of the evaluation criteria and INEX in general is described in [Kazai and Lalmas, 2006].

As discussed in Chapter 2, we use INEX 2005 as a baseline and refer to INEX 2004 results in this part only to explain INEX 2005. That is why we need to discriminate exhaustivity and specificity. Since 2005, specificity has become the focus of INEX evaluations. It was found to reflect the requirements of XML retrieval better.

In order to capture varying degrees of exhaustivity and specificity, INEX has modelled them using graded scales following an investigation by Kekäläinen and Jarvelin [Järvelin and Kekäläinen, 2002]. Some advantages of such a scale are discussed in [Gövert et al., 2006]. Using two measures of relevance, in particular, allows to discuss various degrees of exhaustivity against various degrees of specificity. And, a document component can be compared to its subcomponent. It might be seen to be more exhaustive than its children.

Prior to 2005, INEX has developed a four-point ordinal scale:

1. Not exhaustive (0): The document component information is not about the topic of request (query).
2. Marginally exhaustive (1): The topic of request according to the query is mentioned, but no more than in passing.

3. Fairly exhaustive (2): The document component discusses many aspects about the topic of request of the query, but not all. This includes those requests which have several subtopics and only some of them are considered.

4. Highly exhaustive (3): The document component is fully about the aspects of the query.

For specificity the same principles apply. XML retrieval systems should be rewarded if they deliver focussed document components. A retrieval system that locates the exact relevant paragraph in a document is likely to trigger higher user satisfaction than one that returns a too large component. Again, a binary scale was seen to be not sufficient.

1. Not specific (0): The topic as suggested in the query is not about a theme of the document component.

2. Marginally specific (1): The topic (query) is only a minor theme of the document component.

3. Fairly specific (2): The topic is a mostly covered in the document component.

4. Highly specific (3): The topic is about the document component.

These are the evaluation scales for INEX 2004. INEX 2005 continues to use degrees of exhaustivity and specificity during the evaluation process, but not on an ordinal scale such as the one above. In Section 6.5.3, we discuss the implication in the changes of how the values for exhaustivity and specificity are derived for INEX 2005. Before that, we elaborate the relationship of the evaluation scales of exhaustivity and specificity.

In order to show the relationship between exhaustivity and specificity, we employ the idea of an ideal concept space developed in [Wong and Yao, 1995]. Concepts are the elements in such a concept space, and document components and topics containing concepts are subsets of that concept space. Following this approach in [Gövert et al., 2006], a so-called component coverage matrix is developed that symbolises the differing degrees of overlap in concepts between topic and component for exhaustivity and specificity. This visualisation is very close to aboutness determination, as it treats information represented by a number of concepts as properties of document component and query (topic). The relationship of such concepts in query and documents allows us to determine specificity and exhaustivity.

[Kazai and Lalmas, July 2005] explain specificity and exhaustivity with the ideal concept space. Exhaustivity and specificity can be interpreted with the following formulas: Say T is a topic, C is a component, and $|.|$ is a measure of the size or a counting measure, as van Rijsbergen calls it [van Rijsbergen, 2004] (e.g., the total number of words in a document). Then:

$$exh = \frac{|C \mathrel{\Box\!\!\rightsquigarrow} T|}{|T|}$$

$$spec = \frac{|T \mathrel{\Box\!\!\rightsquigarrow} C|}{|T|}$$

Please note the difference between $C \square\!\!\rightsquigarrow T$ and $T \square\!\!\rightsquigarrow C$, which reflects the difference between exhaustivity and specificity according to Chiaramella's fetch and browse paradigm, as explained in Section 3.3.2.

The concept matrix is a powerful abstraction. It lacks, however, means to represent relationships between the concepts. We would therefore like to reinterpret it as an ideal infon space. As convincing as the abstraction of a concept space for traditional IR seems to be, concepts themselves are not able to express relationships among them. For XML retrieval this is not satisfactory, as structure cannot be represented. The relations between the concepts are neglected in favour of a simplified semantic model. A Situation Theory framework is more powerful. We suggest to use infons instead of concepts in order to include relational infons and therefore structure. With Situation Theory, there is no need to assume independence of the elementary elements.

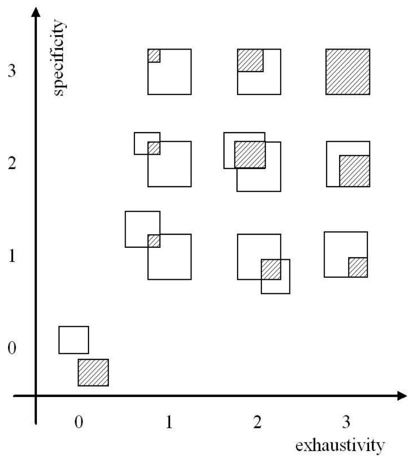


Figure 6.2: Infon coverage matrix with INEX 2004 scale

In the infon coverage Figure 6.2, the upper left square of each entry represents the document component situation, whereas the bottom right square represents the query situation. Together they form an abstract visualisation of an aboutness relation between a query and document component situation. The shaded area symbolises the existence of aboutness. The larger the shaded area the higher the corresponding specificity or exhaustivity value. E.g., a (3,3) combination leads to a full shading, while (2,1) and (2,2) differ in that for (2,1) larger parts of the query situation are not covered by the document component.

Exhaustivity is measured by the size of the overlap of query and document component information in the shaded grey areas. On the other hand, specificity is determined by counting the rest of the information in the component that is not about the query. The less additional, non-useful information can be counted in the component, the higher the specificity value. Thus, specificity measures the relation of relevant to non-relevant content within a single document component.

For the INEX scales, all possible combinations of query and document component situations on the basis of an ideal infon space are shown in Figure 6.2. Each square

Table 6.1: Quantisations in INEX 2004

Function	$f(e,s)$	User model
$Strict_4$	$f(e,s) = \begin{cases} 1 & \text{if e=3 and s=3} \\ 0 & \text{otherwise} \end{cases}$	UU
Gen_4	$f(e,s) = \begin{cases} 1 & \text{if (e,s) = (3,3)} \\ 0.75 & \text{if (e,s) } \in \{(2,3),(3,2),(3,1)\} \\ 0.5 & \text{if (e,s) } \in \{(1,3),(2,2),(2,1)\} \\ 0.25 & \text{if (e,s) } \in \{(1,2),(1,1)\} \\ 0 & \text{if (e,s) = (0,0)} \end{cases}$	EU
SOG	$f(e,s) = \begin{cases} 1 & \text{if (e,s) = (3,3)} \\ 0.9 & \text{if (e,s) = (2,3)} \\ 0.75 & \text{if (e,s) } \in \{(1,3),(3,2)\} \\ 0.5 & \text{if (e,s) = (2,2)} \\ 0.25 & \text{if (e,s) } \in \{(1,2),(3,1)\} \\ 0.1 & \text{if (e,s) } \in \{(2,1),(1,1)\} \\ 0 & \text{if (e,s) = (0,0)} \end{cases}$	SU
$AnyRel$	$f(e,s) = \begin{cases} 0 & \text{if (e,s) = (0,0)} \\ 1 & \text{otherwise} \end{cases}$	TU

represents a situation within a two dimensional space spanned over the three different exhaustivity and specificity values. We have therefore 10 defined positions in this space, as we can discard any combination of (0, i) or (i, 0) with i $\in$ [0, 3]. There can be no specificity without exhaustivity and vice versa.

In Figure 6.2 we can see that the discrimination of scale 1 and 2 for exhaustivity and specificity is based on the relatively larger parts that are not shaded. It is therefore a quantitative difference in degree. We suspect that the discrimination does not add value to an approach investigating aboutness, as it looks at qualitative properties. We shall investigate this in Section 6.5.2.

Figure 6.2 visualises the relationship of the INEX specificity and exhaustivity scales. This chapter considers the changes in the scales used in INEX 2004 and 2005 from a theoretical point of view. Section 6.4 relates them to models for agent reasoning, as they are expressed in the so-called INEX quantisation functions which map the graded scales onto scalar values. Quantisations in INEX reflect the importance attached to exhaustivity and specificity as well as user standpoints as to what constitutes a relevant component [Gövert et al., 2006]. For example, the strict quantisation functions evaluate whether a given retrieval method is capable of retrieving highly exhaustive and highly specific document components.

By representing the agent reasoning in a formal logical framework we will be able to relate them to exhaustivity and specificity. As shown in [Huibers, 1996] rational agents, whether computer systems or human, have the ability to gather information and reason about this gathered information. In Section 6.5, we analyse the aboutness decisions behind the graded scales for INEX 2004 and 2005. We demonstrate how to reason about the changes in the graded scales within our theoretical logic-based framework.

6.4 Agent Representations in INEX Quantisations

In INEX [Gövert et al., 2006] the scales for exhaustivity and specificity are mapped onto ratio scales. To this end, INEX uses quantisation functions over the two parameters of exhaustivity (e) and specificity (s): $f(e, s)$. A single relevance scale [0,1] is the result, as presented in Table 6.1 for INEX 2004 and Table 6.2 for INEX 2005. The first two columns in both tables are taken from [Ogilvie and Lalmas, 2006].

The quantisation functions order the combinations of exhaustivity and specificity values. In Tables 6.1 and 6.2, the strict functions ($Strict_4$ and $Strict_5$) are used to evaluate XML retrieval methods with respect to their capability of retrieving highly exhaustive and highly specific components. The generalised functions (Gen_4 and gen_5) also reward only fairly relevant elements. Other quantisation functions have more specific aims. $AnyRel$ in Table 6.1 evaluates whether INEX approaches can return any relevant element, regardless of their exhaustivity and specificity value. In Table 6.2, $FullySpec$ and $BinExh$ are functions that reward elements independently of exhaustivity. The ? stands for elements that are too small to allow an aboutness conclusion. This value of $f(e, s)$ is new to INEX 2005 reflecting the specific problem with XML document components that are too small to bear information. Further discussion of the quantisations will follow below.

In order to deliver agent representations for the INEX quantisations, we need to express these first in a Situation Theory framework. To do so, we divide the document component and query situations into subsituations, with $D \equiv D_1 \otimes ... \otimes D_n$ and $Q \equiv Q_1 \otimes ... \otimes Q_m$. Please recall that according to Section 3.3.3, a subsituation is a situation S_i that is part of another situation S, where we count the situation as a part of itself, i.e. a situation is a subsituation of itself. Thus, a situation S is about a situation T if and only if T contains a subsituation T_i such that situation S is about situation T_i. Also, we distinguish strict subsituations, i.e. those S_i that are not S.

Using the subsituation-based aboutness criterion from Section 3.3.3, we assume that if D is an exhaustive answer to Q, then it is due to one of the situations D_i that D is composed of. With our subsituation-based aboutness criterion, we are able to represent agent reasoning according to INEX in Section 6.4 and the INEX assessment methodology in Section 6.5 within a single theoretical framework. We speak of rational agent reasoning to include both system and user reasoning.

Quantisations in INEX reflect the importance attached to exhaustivity and specificity. As such they can be used to describe user agent reasoning about results that system agents should return. E.g., $Strict_4$ in Table 6.1 as much as $Strict_5$ in Table 6.2 only credit highly exhaustive and highly specific elements and thus express very demanding user requirements. Within our Situation Theory framework, we have the advantage of being able to express a user's need and a system's attempt to satisfy it within the same framework. Both are reasoning processes that follow rules. This can be considered to be one of the major advantages of a logical theoretical evaluation approach. User assessments are as much as system assessments results of reasoning processes [Huibers, 1996]. In this section, we demonstrate the reasoning of user agents, as we are concerned with the

Table 6.2: Quantisations in INEX 2005

Function	$f(e,s)$	User model
$Strict_5$	$f(e,s) = \begin{cases} 1 & \text{if e=2 and s=1} \\ 0 & \text{otherwise} \end{cases}$	UU
$FullySpec$	$f(e,s) = \begin{cases} 1 & \text{if s = 1} \\ 0 & \text{otherwise} \end{cases}$	SDRU
Gen_5	$f(e,s) = \begin{cases} e * s & \text{if e} \in \{1,2\} \\ 0 & \text{otherwise} \end{cases}$	EU
$GenLifted$	$f(e,s) = \begin{cases} (e+1) * s & \text{if e} \in \{1,2\} \\ s & \text{if e = ?} \\ 0 & \text{otherwise} \end{cases}$	EU
$BinExh$	$f(e,s) = \begin{cases} s & \text{if e} \in \{?,1,2\} \\ 0 & \text{otherwise} \end{cases}$	SDRU

representation of the INEX evaluation methodology.

In the following formalisations D_j and Q_j denote one of n unique subsituations of an XML situation such as an XML element or a query. D_{ex} marks the subsituation that determines a component to be an exhaustive answer, while Q_{sp} states that the component is a specific answer.

The quantisation of $Strict_4$ as much as its INEX 2005 equivalent $Strict_5$ simulates those user agents only interested in highly exhaustive and highly specific answers. These *unanimous* users will only be satisfied if aboutness systems return the highest exhaustivity and specificity values [Huibers, 1996]. The Unanimous User (UU) will only be happy if she can find nothing else, but the two subsituations D_{ex} and Q_{sp}. She wants them to be equivalent to the situations D and Q, respectively, in order to conclude either $D \Box\!\!\leadsto Q$ or $Q \Box\!\!\leadsto D$.

Unanimous User (UU)

$$\frac{D_{ex} \Box\!\!\leadsto Q, D_{ex} \equiv D, Q_{sp} \Box\!\!\leadsto D, Q_{sp} \equiv Q}{D \Box\!\!\leadsto Q, Q \Box\!\!\leadsto D}$$

A user looking for specific answers but at the same time not wanting to entirely lose out on exhaustivity can be called a Specificity-oriented User (SU) represented by SOG in INEX 2004, but without a real equivalent in INEX 2005. SOG only gives *preferences* to specificity by assigning higher quantisation values to higher specificity values.

Specificity-oriented User (SU)

$$\frac{D_1 \boxtimes\!\!\not\leadsto Q, ..., D_n \boxtimes\!\!\not\leadsto Q, Q_{sp} \Box\!\!\leadsto D, Q_{sp} \equiv Q}{D \boxtimes\!\!\not\leadsto Q, Q \Box\!\!\leadsto D}$$

The complement to SOG with a tendency to favouring exhaustivity is Gen_4. It values higher exhaustivity and represents the Exhaustivity-oriented User (EU). As long as most aspects of the query are discussed, the focus is secondary. The Exhaustivity-oriented User (EU) does not neglect specificity fully. The focus, however, is to have $D \Box\!\!\leadsto Q$. For INEX 2005, Gen_5 and $GenLifted$ both place an emphasis on exhaustivity and their Situation

Table 6.3: INEX 2005 exhaustivity and specificity situations

Scale	*Exhaustivity*	*Specificity*
	$D \Box\!\!\rightsquigarrow Q$	$Q \Box\!\!\rightsquigarrow D$
0	$\frac{D_1 \Box\!\!\not\rightsquigarrow Q, ..., D_n \Box\!\!\not\rightsquigarrow Q}{D \Box\!\!\not\rightsquigarrow Q}$	$\frac{Q_1 \Box\!\!\not\rightsquigarrow D, ..., Q_m \Box\!\!\not\rightsquigarrow D}{Q \Box\!\!\not\rightsquigarrow D}$
1	$\frac{D_1 \boxtimes\!\!\not\rightsquigarrow Q}{D \boxtimes\!\!\not\rightsquigarrow Q}, \dots, \frac{D_i \Box\!\!\rightsquigarrow Q}{D \Box\!\!\rightsquigarrow Q}, \dots, \frac{D_n \boxtimes\!\!\not\rightsquigarrow Q}{D \boxtimes\!\!\not\rightsquigarrow Q}$	$\frac{Q_1 \boxtimes\!\!\not\rightsquigarrow D}{Q \boxtimes\!\!\not\rightsquigarrow D}, \dots, \frac{Q_i \Box\!\!\rightsquigarrow D}{Q \Box\!\!\rightsquigarrow D}, \dots, \frac{Q_m \boxtimes\!\!\not\rightsquigarrow D}{Q \boxtimes\!\!\not\rightsquigarrow D}$
2	$\frac{D_1 \boxtimes\!\!\not\rightsquigarrow Q, ..., D_i \Box\!\!\rightsquigarrow Q, ..., D_n \boxtimes\!\!\not\rightsquigarrow Q}{D \Box\!\!\rightsquigarrow Q}$	$\frac{Q_1 \boxtimes\!\!\not\rightsquigarrow D, ..., Q_i \Box\!\!\rightsquigarrow D, ..., Q_m \boxtimes\!\!\not\rightsquigarrow D}{Q \Box\!\!\rightsquigarrow D}$
3	$\frac{D_1 \Box\!\!\rightsquigarrow Q, ..., D_n \Box\!\!\rightsquigarrow Q}{D \Box\!\!\rightsquigarrow Q}$	$\frac{Q_1 \Box\!\!\rightsquigarrow D, ..., Q_n \Box\!\!\rightsquigarrow D}{Q \Box\!\!\rightsquigarrow D}$

Theory representation reflects this by demanding $D \Box\!\!\rightsquigarrow Q$ as an overall conclusion and rewarding those XML elements that include exhaustivity subsituations.

Exhaustivity-oriented User (EU)

$$\frac{D_{ex} \Box\!\!\rightsquigarrow Q, D_{ex} \equiv D, Q_1 \boxtimes\!\!\not\rightsquigarrow D, ..., Q_n \boxtimes\!\!\not\rightsquigarrow D}{D \Box\!\!\rightsquigarrow Q, Q \boxtimes\!\!\not\rightsquigarrow D}$$

In INEX 2004, the *AnyRel*-function captures the typical user of mass information systems, happy with any relevant component. There is no equivalent in INEX 2005. The Typical User (TU) would like to see any kind of subsituations, allowing to conclude either exhaustivity or specificity. She is not interested in an overall conclusion of $D \Box\!\!\rightsquigarrow Q$ or $Q \Box\!\!\rightsquigarrow D$, but in partial conclusions indicating either an exhaustive or a specific answer.

Typical User (TU)

$$\frac{D_1 \Box\!\!\rightsquigarrow Q}{D \Box\!\!\rightsquigarrow Q}, \dots, \frac{D_n \Box\!\!\rightsquigarrow Q}{D \Box\!\!\rightsquigarrow Q}, \frac{Q_1 \Box\!\!\rightsquigarrow D}{Q \Box\!\!\rightsquigarrow D}, \dots, \frac{Q_n \Box\!\!\rightsquigarrow D}{Q \Box\!\!\rightsquigarrow D}$$

Instead of a direct equivalent to SU, INEX 2005 comes up with two new user types $BinExh$ and $FullySpec$. Both only look for specificity, as long as exhaustivity is not impossible. $BinExh$ is not as strict with respect to the exhaustivity value. In this sense, it corresponds to Chiaramella's earlier suggestions that describe the focus of the answer as the specific interest of XML retrieval. [Chiaramella, 2001] has demonstrated within a theoretical experiment that Structured Document Retrieval Users (SDRU) are interested in specificity as long as the answer remains exhaustive enough. This is why we call this model SDRU:

Structured Document Retrieval User (SDRU)

$$\frac{D_1 \boxtimes\!\!\not\rightsquigarrow Q, ..., D_n \boxtimes\!\!\not\rightsquigarrow Q, Q_{sp} \Box\!\!\rightsquigarrow D, Q_{sp} \equiv Q}{Q \Box\!\!\rightsquigarrow D}$$

SDRU's differ from SU's only in that their overall conclusion is only influenced by specificity. SDRU's are looking to find a $Q_{sp} \Box\!\!\rightsquigarrow D$ in order to conclude $Q \Box\!\!\rightsquigarrow D$. Not all users of XML retrieval systems have to be SDRU's, but the particular interest of XML

Table 6.4: INEX 2004 exhaustivity and specificity situations

Scale	*Exhaustivity* $D \Box\!\!\rightsquigarrow Q$	*Specificity* $Q \Box\!\!\rightsquigarrow D$
0	$\dfrac{D_1 \Box\!\!\not\rightsquigarrow Q, ..., D_n \Box\!\!\not\rightsquigarrow Q}{D \Box\!\!\not\rightsquigarrow Q}$	$\dfrac{Q_1 \Box\!\!\not\rightsquigarrow D, ..., Q_n \Box\!\!\not\rightsquigarrow D}{Q \Box\!\!\not\rightsquigarrow D}$
1	$\dfrac{D_1 \boxtimes\!\!\not\rightsquigarrow Q, ..., D_n \boxtimes\!\!\not\rightsquigarrow Q}{D \Box\!\!\rightsquigarrow Q}$	$\dfrac{Q_1 \boxtimes\!\!\not\rightsquigarrow D, ..., Q_n \boxtimes\!\!\not\rightsquigarrow D}{Q \Box\!\!\rightsquigarrow D}$
2	$\dfrac{D_1 \boxtimes\!\!\not\rightsquigarrow Q, ..., D_n \boxtimes\!\!\not\rightsquigarrow Q, D_{ex} \Box\!\!\rightsquigarrow Q}{D \Box\!\!\rightsquigarrow Q}$	$\dfrac{Q_1 \boxtimes\!\!\not\rightsquigarrow D, ..., Q_n \boxtimes\!\!\not\rightsquigarrow D, Q_{sp} \Box\!\!\rightsquigarrow D}{Q \Box\!\!\rightsquigarrow D}$
3	$\dfrac{D_{ex} \Box\!\!\rightsquigarrow Q, D_{ex} \equiv D}{D \Box\!\!\rightsquigarrow Q}$	$\dfrac{Q_{sp} \Box\!\!\rightsquigarrow D, Q_{sp} \equiv Q}{Q \Box\!\!\rightsquigarrow D}$

retrieval compared to flat document retrieval is better represented by SDRU's than by other agent models, as the overall conclusion is focussed on specificity only.

To better see the overall use of these agent reasoning models, let us briefly investigate what is possible if we can express system and user reasoning in the same framework. We can combine, for instance, Left Monotonic Union (LMU) and Unanimous User (UU) model in:

$$\textbf{UU:}\ \frac{D_{ex} \Box\!\!\rightsquigarrow Q, D_{ex} \equiv D}{D \Box\!\!\rightsquigarrow Q}$$

$$\textbf{LMU:}\ \frac{D \Box\!\!\rightsquigarrow Q}{D \otimes D1 \Box\!\!\rightsquigarrow Q}$$

$$\Rightarrow \frac{D \otimes D1 \Box\!\!\rightsquigarrow Q}{D \otimes D1 \not\equiv D_{ex}}$$

The conclusion that $D \otimes D1 \Box\!\!\rightsquigarrow Q$ clearly contradicts the assumption of the UU that $D_{ex} \equiv D$, which means UU's will not be served well by aboutness reasoning systems that include LMU.

In this section, we have presented agent reasoning models, as expressed in the INEX quantisations for XML retrieval, based on Chiaramella's differentiation of $D \Box\!\!\rightsquigarrow Q$ and $Q \Box\!\!\rightsquigarrow D$. We have added a third column to Tables 6.1 and 6.2 to summarise these results. We have shown the new focus in INEX 2005 on specificity and would like to investigate this issue further by looking at the transition in terms of the system agents' rewards from INEX 2004 to INEX 2005.

The next section places the INEX exhaustivity and specificity assessment scales into the context of Situation Theory. We will show that system agents are rewarded if they reflect the user agent reasonings. For example, in order to reach the highest values for exhaustivity and specificity, they must support the reasoning of unanimous users.

6.5 Exhaustivity and Specificity Assessments in INEX 2004 and 2005

We start by presenting the reasoning behind the INEX 2004 and 2005 relevance scales and user models within a Situation Theory framework, to afterwards relate the user models and quantisations to this reasoning. We show that exhaustivity and specificity are two sides of the same aboutness relation and not two independent criteria. We follow a similar argument as in [Ogilvie and Lalmas, 2006], where the authors argue for a focus on specificity, as this is the specific interest in XML retrieval and sufficient to evaluate it. Since INEX 2006 only specificity has been used to measure retrieval effectiveness.

We continue our modelling of how an agent perceives how exhaustive and specific a component is. We argue that an agent, either a system or a user, assesses the relevance of a component according to the information contained in both Q and D. In the next two subsections, we develop two tables representing reasoning models according to INEX 2004 and 2005 definitions of graded scales of relevance, respectively. In the third subsection, we explore the relationship of exhaustivity and specificity for INEX 2005 and argue for a better integrated assessment showing that, for both specificity and exhaustivity, it is the same relevant information that decides on its values.

6.5.1 INEX 2004 Reasoning

We aim to show how INEX 2004 reasoning is based on the subsituation-based aboutness decision from Section 3.3.3. In order to do so, we start discussing the INEX 2004 exhaustivity and specificity situation on the background of the liberal aboutness criterion by Huibers and Bruza first. Afterwards we demonstrate that we achieve a more consistent view with a subsituation-based aboutness decision. We build upon work in [Blanke and Lalmas, 2006] that has used Situation Theory to formally represent assessment decisions in INEX 2004.

Table 6.3 demonstrates our translation of Figure 6.2 into INEX 2004 exhaustivity and specificity situations. It summarises aboutness decisions for INEX 2004 with a liberal aboutness decision demanding any common information to derive aboutness. We argue that a user assesses the relevance of a component according to the information contained in both Q and D.

In Table 6.3, the document component and query situations are divided into other subsituations, with $D \equiv D_1 \otimes ... \otimes D_n$ and $Q \equiv Q_1 \otimes ... \otimes Q_n$. Scale 1 states that only some of its subsituations are about the query but none involves anti-aboutness. With this, we can, e.g., formalise what is meant by a marginally exhaustive document component (1): the topic is only mentioned in passing, leading to very small indications about the document components relevance.

Table 6.3 shows that for scale 1 multiple conclusions are possible, demonstrating undecidedness about the component's relevance. For scale 2, the overall conclusion can be derived that $D \Box\!\!\rightsquigarrow Q$ or $Q \Box\!\!\rightsquigarrow D$. For specificity, the topic is a major theme of the document component and $Q \Box\!\!\rightsquigarrow D$ can be concluded. Scale 0 indicates that no D_i is about the query making the whole document component not about the query. The highest satisfaction is achieved with scale 3. For exhaustivity, all subsituations of the component are about the query, while for specificity all subsituations of the query are about the document component.

A combination of (0,3), e.g., is impossible, as our perception of exhaustivity and specificity is based on the overlap between the query and document component situation according to Figure 6.2 and the shaded areas in it. With $D \equiv map(A)$ and $Q \equiv map(B)$, the conclusions of $D \Box\!\not\leadsto Q$ (and therefore $A \cap B \equiv \emptyset$) and $Q \Box\!\leadsto D$ (and therefore $B \cap A \not\equiv \emptyset$) contradict each other. The same argument obviously applies to all combinations of exhaustivity and specificity where one has the value 0 and the other has not. The infon coverage matrix 6.2 is right to exclude these.

Table 6.4 describes the INEX 2004 reasoning with a subsituation-based aboutness criterion. For scale 0, we cannot find any subsituation that would justify an aboutness conclusion. Scale 1 states that we are undecided whether we can call this aboutness. For this scale, there is no strict subsituation that would allow us to conclude aboutness. For scale 2, D_{ex} is a strict subsituation that makes D exhaustive, while the rest of the subsituations of D are numbered from 1 to n. For 3, there is no other information in the assumption than the subsituation having the property exhaustivity or specificity. Users expect to see only information that is relevant, i.e. $Q_{sp} \equiv Q$ for specificity and $D_{ex} \equiv D$ for exhaustivity.

Table 6.4 corresponds to the in Section 6.4 defined user agent models. An UU agent model expects a system agent to react only to the assumptions of no other information than the relevant one for both exhaustivity and specificity. In order to support a SU, a value of at least 1 for exhaustivity is required to be delivered by the system agent. Within the reasoning of the system agent, the assumptions $D_1 \boxtimes\!\not\leadsto Q, ..., D_n \boxtimes\!\not\leadsto Q$ must be achieved. Exhaustivity must not have the value 0, as this would make specificity have a value of 0 as well. Analogously, for a system agent to reflect an EU the specificity value counts only as far as it is not 0.

The TU is not really represented in the INEX 2004 exhaustivity and specificity situations. Her conclusions are binary and not scaled — either an answer is relevant or not. Therefore the non-representation of the typical user does not affect our representation and is rather an indication that it is separate from the other INEX agent reasoning models. The typical user does not appear again in later INEX assessments. For a logical reasoning model also difficult to discriminate are degrees of reasoning. In particular, without extension to our Situation Theory framework it seems impossible to discriminate an assessment of 1 or 2. We will later see that this does not pose a problem, as for INEX 2005 assessments the two middle-valued assessments are merged into one.

Let us briefly discuss examples of how such agent reasonings behind the INEX 2004 scales are reflected in combined exhaustivity and specificity assessments. For demonstration purposes, we focus on combinations of very high expectations for specificity or exhaustivity. A combined assessment of (3,3) clearly means an exact match, as $D_{ex} \equiv D$ according to the exhaustivity judgment's assumptions, as well as $Q_{sp} \equiv Q$ according to the specificity

Table 6.5: INEX 2005 exhaustivity and specificity situations

E-Scale	*Exhaustivity* $D \square\!\!\rightsquigarrow Q$	*Specificity* $Q \square\!\!\rightsquigarrow D$	*S-Scale*
0	$\frac{D_1 \square\!\!\not\rightsquigarrow Q, ..., D_n \square\!\!\not\rightsquigarrow Q}{D \square\!\!\not\rightsquigarrow Q}$	$\frac{Q_1 \square\!\!\not\rightsquigarrow D, ..., Q_n \square\!\!\not\rightsquigarrow D}{Q \square\!\!\not\rightsquigarrow D}$	0
1	$\frac{D_1 \boxtimes\!\!\not\rightsquigarrow Q, ..., D_n \boxtimes\!\!\not\rightsquigarrow Q, D_{ex} \square\!\!\rightsquigarrow Q}{D \square\!\!\rightsquigarrow Q}$	$\frac{Q_1 \boxtimes\!\!\not\rightsquigarrow D, ..., Q_n \boxtimes\!\!\not\rightsquigarrow D, Q_{sp} \square\!\!\rightsquigarrow D}{Q \square\!\!\rightsquigarrow D}$	$\frac{\lvert Q_{sp}\rvert}{\lvert D\rvert}$
2	$\frac{D_{ex} \square\!\!\rightsquigarrow Q, D_{ex} \equiv D}{D \square\!\!\rightsquigarrow Q}$	$\frac{Q_{sp} \square\!\!\rightsquigarrow D, Q_{sp} \equiv Q}{Q \square\!\!\rightsquigarrow D}$	1

judgment's assumptions. Furthermore, (3,2) implies that the subsituation Q_{sp} must be about the subsituation D_{ex}: $Q_{sp} \square\!\!\rightsquigarrow D_{ex}$, with $Q_{sp} \square\!\!\rightsquigarrow D$ according to the assumptions about a scale 2 specificity judgment and $D_{ex} \equiv D$ according to full exhaustivity. This insight is confirmed by an example, where the query is $\{\langle\langle house\rangle\rangle\}$ and $\{\langle\langle garden\rangle\rangle\}$, while the document component has $\{\langle\langle house\rangle\rangle\}$, $\{\langle\langle garden\rangle\rangle\}$ and $\{\langle\langle car\rangle\rangle\}$. For this example $D \equiv D_{ex} \otimes D_1$, with $D_{ex} \equiv \{\langle\langle house\rangle\rangle, \langle\langle garden\rangle\rangle\}$ and $D_1 \equiv \{\langle\langle car\rangle\rangle\}$. Thus, the subsituation Q_{sp} must be about D_{ex} and must be $\{\langle\langle house\rangle\rangle, \langle\langle garden\rangle\rangle\}$. A combined assessment of (1,3) implies that none of the other subsituations of the exhaustivity judgment contradicts the information in Q_{sp}, with $D_k \boxtimes\!\!\not\rightsquigarrow Q$ and $Q_{sp} \equiv Q$.

In this subsection, we have shown how a subsituation-based aboutness criterion can be used to formalise INEX 2004 assessment decisions. In the next subsection, we present the transition from INEX 2004 to INEX 2005, where the focus is much more on specificity.

6.5.2 INEX 2005 Reasoning

According to several studies investigating agreements in the relevance assessments, e.g. [Trotman, 2005], the discrimination of scale 1 and 2 in INEX 2004 does not add value and could potentially lead to confusion. This is also confirmed from a subsituation-based aboutness point of view. For INEX 2004 in Table 6.4, either a subsituation Q_{sp} or D_{ex} exists (scale 2 and scale 3) or not (scale 0 and scale 1) and if it exists it is either a strict subsituation (scale 2) or the complete situation (scale 3). Three-valued scales cover all the differences in a subsituation-based aboutness, with 0 meaning no subsituation for relevance exists, 1 meaning a strict subsituation exists and 2 meaning the complete XML situation is relevant. INEX 2005 [Ogilvie and Lalmas, 2006] has such a three-valued scale for exhaustivity.

For specificity, a continuous scale is applied in INEX 2005 with values in [0,1], where 1 represents a fully specific component. The specificity value is derived as follows: In a first phase, assessors highlight text fragments containing relevant information, so that in the end each XML element has highlighted parts and non-highlighted parts. In a second step, the ratio of highlighted text and total text per XML element delivers a specificity value between 0 and 1. This procedure was adopted following the outcome of studies showing it to be a more natural way for assessing relevance with more consistent assessments [Pehcevski and Thom, 2006]. This new procedure is interesting, as for the first time in INEX, it fulfills the inherent mathematical relationship between query and document component, as visualised in Figure 6.2 and used to define pure type aboutness in Section 4.7.1.

The new INEX 2005 specificity measure was developed according to the formalism in [Gövert et al., 2006] describing specificity as the focus of a document component on the topic in the query. This focus is exemplified by using the relationship of the size of those parts of a document component that are about the query to the size of those that are not about the query. [Gövert et al., 2006] describes specificity as the relationship of a topic and a component. As for the aboutness relation a topic is in IR defined by the query, we can use the relationship of query and document component as a specificity measure. Furthermore, instead of using concepts as the carriers of meaning as in [Gövert et al., 2006], we use situations. To do so, we first need a measure for the size of a situation's information.

Let $|S|$ represent such a measure of the information size of a situation S. In INEX 2005, assessors highlight those parts of the document component that are about the query. In INEX 2005 characters are counted to determine the length of highlighted text and its relation to the total size of the element. By using the character ratio of highlighted and non-highlighted text for the specificity judgment, INEX 2005 demands that the specificity value should be a direct reflection of the counting size of the aboutness situation in relation to the document component situation: $spec = \frac{|Q \Box\!\!\rightsquigarrow D|}{|D|}$.

By committing itself at least for specificity to a strict relationship between highlighted and non-highlighted text, INEX 2005 uses the extent to which the query topic is a subset of the component topic as an aboutness criterion for the specificity assessment. Therefore, the highlighting of the assessors forms an aboutness reasoning, where the highlighted parts describe the subsituation Q_{sp} that makes the document component a specific answer. Thus, the size of $|Q \Box\!\!\rightsquigarrow D|$ is identical to the size of Q_{sp} producing specificity, as Q_{sp} describes exactly those parts of a document component that are making a component relevant to the query: $spec = \frac{|Q_{sp}|}{|D|}$. Obviously, the fraction would be 0, if Q_{sp} would not exist or 1 if $Q_{sp} \equiv D$.

Table 6.5 summarizes the way INEX 2005 assigns values to agent reasoning. The specificity value is directly linked to the difference of $|Q_{sp}|$ and $|D|$. Regarding exhaustivity, we follow the above explained logic of subsituation-based aboutness. Then, 0 expresses that we cannot find a subsituation to conclude exhaustivity. For scale 1 such a subsituation exists and for scale 2 D_{ex} is the complete situation. As an example for a combined assessment, the new user type in INEX 2005, the SDRU would like to see that $D_{ex} \Box\!\!\rightsquigarrow Q_{sp}$, with a specificity value of 2 requiring $Q_{sp} \equiv Q$ and an exhaustivity value of at least 1 demanding $D_{ex} \Box\!\!\rightsquigarrow Q$. $D_{ex} \Box\!\!\rightsquigarrow Q_{sp}$ is a requirement to satisfy the SDRU, which proves Chiaramella's assumption: In order to achieve the best focus for answers, we have to choose from those XML elements that are exhaustive answers. Their subsituations must be about the specificity subsituation.

Even more than Table 6.4, Table 6.5 is derived from the above described agent reasoning models, as here the three-scaled assessments correlate to the idea of subsituation-based aboutness. The UU is still best represented in system agents that deliver just relevant

Table 6.6: Example with new exhaustivity and specificity measures

	Q1		Q2		Q3		Q4		Q5	
	spec	exh	spec	exh	spec	exh	spec	exh	spec	exh
D1	0.73	1	0.33	1	0.4	1	1	1	1	0.79
D2	0.31	0.45	0.31	1	0	0	0.31	0.33	0.31	0.33

information and nothing else but non-strict subsituations for specificity and exhaustivity. The EU is still favoured by system agents that avoid the value 0 for specificity, but whose reasoning demands at least $Q_1 \boxtimes\not\rightsquigarrow D, ..., Q_n \boxtimes\not\rightsquigarrow D, Q_{sp} \Box\rightsquigarrow D$. In INEX 2005, it becomes clear that this implies finding any kind of subsituation with the exhaustivity property. Lastly, the SDRU demands from system agents any kind of subsituation with the exhaustivity property allowing her to focus on the conclusion of specificity. Contrary to INEX 2004, all agent models are covered in Table 6.5 and no problems occur in discriminating the different degrees.

Theoretically, we have demonstrated in detail that specificity and exhaustivity are not independent values while discussing the fetch and browse paradigm in Chapter 2. Both XML evaluation dimensions can only be discriminated by aboutness systems that can distinguish the left and right hand side of an aboutness relation. An example for such a relation would be the subset relation, as used in the INEX 2005 assessments or pure type aboutness.

To recall, the representation of specificity as $Q \Box\rightsquigarrow D$ originates from the fetch and browse paradigm of Chiaramella. The specificity decision is directly dependent on the exhaustivity one, as first $D \Box\rightsquigarrow Q$ is evaluated and from this set the most specific information can be found by allowing only $Q \Box\rightsquigarrow D$. This paradigm already shows that exhaustivity and specificity are not two independent values, but rather two dimensions of the same relation. That, according to Chiaramella, there is no specificity without exhaustivity, makes us suspect that we might be able to drop one of the dimensions for the evaluation — not because it is not analytically interesting, but because it might not add value to the aboutness decision. We analyse this question in the next section by investigating the use value of the different quantisation functions in INEX 2005. Is the separate assessment of exhaustivity and specificity necessary? After all, specificity is the focus of XML retrieval. Structure is used to add focus to the general retrieval task.

In the next subsection, we investigate one of the reasons why INEX 2006 has decided to focus only on specificity in order to evaluate retrieval effectiveness. We show that, according to INEX 2005, specificity and exhaustivity values are inseparable and represent two views of the same relationship of query and XML element. This becomes clear as a result of INEX 2005, because specificity follows such a well-defined assessment strategy.

Another paper stressing the primary importance of specificity for XML retrieval from a completely different angle than Chiaramella is [Ogilvie and Lalmas, 2006]. In this paper, Ogilvie and Lalmas use extensive statistical tests to perform an analysis of the exhaustivity and specificity dimensions used in two rounds of INEX. Their conclusion is that specificity is a sufficient evaluation dimension for XML retrieval. We arrive at similar conclusions from a theoretical perspective in the next section.

6.5.3 The Relation of Exhaustivity and Specificity in INEX

First, we would like to show that with respect to the different quantisations in INEX 2004 and 2005 exhaustivity and specificity are not two independent values. If we say in agreement with the judgment correcting the INEX 2004 scale that we can ignore all those judgments involving a 1 in INEX 2004, we can reduce the complexity of Gen_4 and SOG radically. Furthermore, we ignore the extreme cases of a (3,3) or (0,0) assessment, as they do not differ in Gen_4 and SOG. What remains is the following combination of scale 2, scale 3, and (2,2) assessments:

$$f_{gen_4}(e,s) = \begin{cases} 0.75 & \text{if (e,s)} \in \{(2,3),(3,2),(3,1)\} \\ 0.5 & \text{if (e,s)} \in \{(1,3),(2,2),(2,1)\} \end{cases}$$

$$f_{SOG}(e,s) = \begin{cases} 0.9 & \text{if (e,s)} = (2,3) \\ 0.75 & \text{if (e,s)} \in \{(1,3),(3,2)\} \\ 0.5 & \text{if (e,s)} = (2,2) \end{cases}$$

Thus, the specificity-oriented f_{SOG} and the exhaustivity-oriented f_{gen_4} differ only in that f_{SOG} rewards (2,3) more than (3,2), as it focuses on specificity. We would like to argue that for most retrieval models this does not deliver the discrimination desired. We confirm the observation in [Ogilvie and Lalmas, 2006] that both quantisations behave similarly when ranking systems. At least, they do not have the strong discriminatory effect hoped for.

We use our Situation Theory framework to show that the reasoning for (2,3) and (3,2) does not differ for most cases of aboutness systems. According to Table 6.4 (3,2) leads to the conclusion that $D_{ex} \Box\!\!\rightsquigarrow Q_{sp}$. The subsituation making $D \Box\!\!\rightsquigarrow Q$ must be about the subsituation making the aboutness relation specific. Analogously, for (2,3) $D_{ex} \Box\!\!\rightsquigarrow Q_{sp}$ holds. Thus, it is only for models based on aboutness decisions that can discriminate $D_{ex} \Box\!\!\rightsquigarrow Q_{sp}$ and $Q_{sp} \Box\!\!\rightsquigarrow D_{ex}$, that these two values really make a difference.

Most INEX models rely on information overlap to decide aboutness, as seen in Chapter 5. As also seen in the chapter, information-overlap based systems do not support Symmetry and therefore do not discriminate $D_{ex} \Box\!\!\rightsquigarrow Q_{sp}$ from $Q_{sp} \Box\!\!\rightsquigarrow D_{ex}$. Overlap is the basis of many successful models like the vector space models, etc. That these models are also used for XML retrieval, is one reason why different quantisations for (2,3) and (3,2) do not deliver a significant difference in ranking systems.

[Ogilvie and Lalmas, 2006] also conclude that the model differences identified by the $AnyRel$ quantisation tend to be also identified by the Gen_4 and SOG functions. Please recall that in Table 6.4 a value of 2 for exhaustivity or specificity means that a subsituation allowing these judgments can be found, but that there are other subsituations. These subsituations seem to play only a secondary role for the agents' assessments. We have seen in the previous subsection that the subsituation deciding specificity is exactly defined in INEX 2005 rather than being the result of a more intuitive agent decision. It was taken out of the hand of the user to decide on degrees after studies that showed highlighting to be a more natural way for assessors to decide on degrees [Kazai and Lalmas, 2005].

The question of the relation of exhaustivity and specificity values for INEX has been intensely discussed. [Ogilvie and Lalmas, 2006] go as far as to suggest that after INEX 2005 only specificity should be used for the evaluation, as the specificity-oriented quantisation $BinExh$ can be used to predict exhaustivity-oriented quantisations such as $GenLifted$. This indicates that exhaustivity and specificity are not independent values. Additionally, [Chiaramella, 2001] declares that a specificity judgment should be seen as a more narrow focus of an exhaustivity assessment. As discussed in Section 3.3.1, in his theoretical framework, $Q \Box\!\!\rightsquigarrow D$ is only evaluated against those document component situations D that are about the query Q: $D \Box\!\!\rightsquigarrow Q$. We cannot go as far as Chiaramella. By validating our assumption that the same relevant information is used for the exhaustivity and specificity assessment within an example, we can, however, state that these two assessment values offer two views on the same aboutness relation between query and document component. We need to look at both evaluation measures in terms of one being the condition of the other. In particular, without exhaustivity no specificity aboutness.

From this point of view, it is interesting to see how the INEX 2005 quantisations behave. In particular, the above quoted relationship of the exhaustivity-oriented quantisations Gen_5 and $GenLifted$ to $BinExh$ is telling, as the latter exactly expresses the above condition: Without exhaustivity no specificity, but specificity delivers the end result. The other two reward higher exhaustivity values by multiplying a variant of that value with the specificity value for the overall result. Again, we can say that those models able to deliver D_{ex} are favoured. Yet, we have just argued that most of these models are at the same time able to deliver Q_{sp}, as overlap between information is such an important feature of aboutness systems according to Chapter 5. Also, as shown, the assessors orientate themselves on the overlap of query and document situation while highlighting relevant document parts. Therefore, the discriminative power of the quantisations Gen_5 and $GenLifted$ is not much better than $BinExh$, though their absolute results might be higher. They value the same aboutness relations with higher values.

Thus, [Ogilvie and Lalmas, 2006] rightly contest the value of a graded exhaustivity value for INEX 2005. A graded scale does not seem to be necessary to treat exhaustive elements as relevant. Looking at $Strict_5$ in the INEX 2005 quantisations, there is no way for us to include it in other quantisations. Its exhaustivity assumption is too strict. It only considers fully exhaustive elements. Yet, [Ogilvie and Lalmas, 2006] argue that $Strict_5$ has anyway not convinced with its ability to discriminate systems and should be discarded for future evaluations. Let us now look at an alternative to a scaled exhaustivity measure, an alternative which is close to the way specificity is already measured.

As exhaustivity and specificity in INEX express the relationship between one document component and one query, they bear a direct mathematical interpretation, which have used to define pure type aboutness. Within our Situation Theory framework, the factor by which one document component covers only aspects of one query (specificity), and the factor by which one document component is about all aspects of a query (exhaustivity), can be interpreted as follows: Say Q is a query situation, D is one given component situation, and $|D \Box\!\!\rightsquigarrow Q|$ stands for the degree/size to which D is about Q. In INEX 2005, this could be measured using the same character counting method, as used for specificity (see Section 6.5.2). In fact, as the information determining exhaustivity and specificity is the same, the same characters can be used. We shall discuss this in more detail later. According to the formalism in [Gövert et al., 2006], exhaustivity can be defined as the degree to which all aspects of the query are covered by comparing the size of the parts of the aboutness relation that make the document component an exhaustive answer with the size of the query topic: $exh = \frac{|D \Box\!\!\rightsquigarrow Q|}{|Q|}$. As shown specificity is $spec = \frac{|Q_{sp}|}{|D|}$.

The relevant information in the document component is the same for exhaustivity and specificity. Exhaustivity and specificity only differ in representing whether this relevant information covers all aspects of the topic for exhaustivity or whether for specificity the relevant information does not come with irrelevant information in the same document component. Therefore the highlighting of the INEX 2005 assessors for specificity will also have identified the parts of the document components that determine how exhaustively it answers to the topics in the query. Highlighting is about what is relevant. Specificity and exhaustivity are only two different views on how this relevant information relates to other information — either in the query or in the document component.

As already discussed, in INEX 2005 the exhaustivity value is independently chosen from the specificity one.[1] It is not formally based on highlighting. For exhaustivity, assessors are 'free' to choose a value between 0 and 2, while specificity is determined by calculating the relation of highlighted to non-highlighted text. The deliberation that the relevant information stays the same offers a mathematical relationship of the INEX 2005 specificity value towards exhaustivity. Instead of judging exhaustivity without using the highlighted text, an alternative idea for exhaustivity would be to use the second formula from [Gövert et al., 2006]: $exh = \frac{|D \Box\!\!\rightsquigarrow Q|}{|Q|}$, as for specificity the complementary equation $spec = \frac{|Q_{sp}|}{|D|}$ is used. Looking at it from a subsituation-based aboutness point of view, exh has a clear mathematical interpretation similarly to $spec$ by relating the counting size of the exhaustivity subsituation to the counting size of the query: $exh = \frac{|D_{ex}|}{|Q|}$. This relation measures the degree to which a document component covers the concepts requested by a topic.

In the following paradigmatic example, using $exh = \frac{|D_{ex}|}{|Q|}$ and $spec = \frac{|Q \Box\!\!\rightsquigarrow D|}{|D|}$ as evaluation measures, we demonstrate that the results are the expected preferences in terms of exhaustivity and specificity. To keep the aboutness relation simple, we assume that the information in query and document components is constituted by their keywords and by their keywords only. Furthermore, we assume that the assessors only use the overlap between these keywords as an aboutness decision. A document component stating 'Dogs are not cats' is relevant to a query about the topic 'cats', as the keyword cats is part of the document component. This keyword will be the only highlighted part in the component. In a Situation Theory framework the document component is $\{\, \langle\langle dogs \rangle\rangle \,,\, \langle\langle cats \rangle\rangle \,\}$ and

[1]The exception is non-aboutness: A value of 0 for specificity has to mean a 0 for exhaustivity and vice versa.

the query is $\{ \langle\langle cats \rangle\rangle \}$. The highlighted text will have 4 characters, the non-highlighted 13 characters. In order to simplify our calculations and keep the overview in our example, we assume that the document components only consist of keywords. Then the counting size of a document situation in the INEX 2005 assessment will be the number of characters each keyword information item has plus one whitespace separating the keywords in the text. Like this we avoid confusion about the counting size when describing the document components and query in our example as situations.

Let us therefore assume the following assessment situation: $\{ \langle\langle house \rangle\rangle , \langle\langle garden \rangle\rangle , \langle\langle flat \rangle\rangle \}$ is document component situation $D1$ with a counting size of 15. Let $D2$ be $\{ \langle\langle house \rangle\rangle , \langle\langle close \rangle\rangle , \langle\langle garage \rangle\rangle \}$ with $|D2| = 16$. We have 5 query situations: $Q1$ is $\{ \langle\langle house \rangle\rangle , \langle\langle garden \rangle\rangle \}$, $Q2$ $\{ \langle\langle house \rangle\rangle \}$, $Q3$ $\{ \langle\langle garden \rangle\rangle \}$, $Q4$ $\{ \langle\langle house \rangle\rangle , \langle\langle garden \rangle\rangle , \langle\langle flat \rangle\rangle \}$, and $Q5$ $\{ \langle\langle house \rangle\rangle , \langle\langle garden \rangle\rangle , \langle\langle flat \rangle\rangle , \langle\langle car \rangle\rangle \}$. Then, $|Q1| = 11$, $|Q2| = 5$, $|Q3| = 5$, $|Q4| = 15$ and $|Q5| = 19$. This example is paradigmatic, as we cover all 4 possible combinations following Figure 6.2: either the information in the query is fully covered in the document component, or the document component has no other, but not all information of the query, or both document component and query share information, but both also have other information, or query and document component do not share information at all.

Table 6.6 summarizes the assessment outcomes for $exh = \frac{|D_{ex}|}{|Q|}$ and $spec = \frac{|Q_{sp}|}{|D|}$ in this example. It clearly presents the expected preferences in terms of exhaustivity and specificity assessments. Also, a 0 assessment in one of the measures leads to a 0 assessment in the other. The example of $Q4$ and $D1$ offers a complete match. The assessment results are always between 0 and 1, as the size of the information overlap of query and document component can never be larger than either the counting size of the query situation or the counting size of the component situation. An empty query or an empty document component will lead to an undefined assessment result. For XML retrieval, an empty document component is an XML element without content, which we defined above as meaningless and therefore no subsituation. We could at this point easily introduce a threshold for the counting size of the subsituation that would exclude those that are too small and therefore meaningless, as it has been done for INEX 2005. Finally, Table 6.6 shows that it is possible to use the same relevant information as a basis for the exhaustivity and specificity assessments without changing their outcomes in the INEX 2005 assessment procedure. This supports Chiaramella's idea of specificity and exhaustivity as two dependent values.

6.6 Conclusion

In this chapter, we have delivered a new perspective deriving from the possibilities of theoretical evaluation. We have presented a theoretical evaluation to evaluate existing experimental evaluations. Rational agents, whether computer systems or humans, have the ability to gather information and reason about this gathered information. We use our Situation Theory framework to represent an information need and a system's evaluation of it as a result of reasoning processes, because Situation Theory does not just apply to machine reasoning but also human reasoning. This can be considered to be one of the major advantages of a logic-based theoretical evaluation approach.

Finally, our theoretical evaluation of the INEX experimental evaluation expresses specificity and exhaustivity as properties of aboutness. If we consider exhaustivity and specificity to be properties of situations, we arrive at a new way of determining the INEX 2005 exhaustivity dimension. We can apply the same mathematical rigour to it as has been done for INEX 2005 specificity. We suggested that $exh = \frac{|D_{ex}|}{|Q|}$ is a complementary measure to specificity defined by $spec = \frac{|Q_{sp}|}{|D|}$ for INEX 2005.

As a result, we conclude that it is possible to use a theoretical evaluation to evaluate experimental evaluations. This is particularly true for the case of INEX, where we have a well-defined mathematical relation for the main evaluation measure of specificity. The next chapter introduces another new dimension of a theoretical evaluation if we use Situation Theory to analyse specificity aboutness.

Chapter 7

Determining Specificity Aboutness

7.1 Introduction - Being Specific

In INEX the retrieval task that aims at finding the most specific answers has been referred to as the focussed task [Kazai and Lalmas, 2005]. This is to be compared to the thorough task, that aims at estimating the relevance of document components to a query. In this latter task, all relevant document components are to be identified, and then ranked according to their degree of relevance. In the focussed task, the result set should consist of non-overlapping document components, ranked according to how specific they are to the query.

As discussed in Section 4.7.1, overlap occurs if a document component (e.g. a section) and one of its descendents (e.g. a paragraph in this section) or ascendents (e.g. the chapter containing that section) are both returned as answers. The aim of the focussed task is therefore to identify the component that is the most specific to the query among overlapping document components, and to return it as what is referred to as a *focussed* answer.

INEX models to implement the focussed retrieval task can be viewed as filters. Indeed, these models mostly consist of the post-processing of an answer set produced by models aimed at implementing the thorough retrieval task. The post-processing phase consists of eliminating all but the most focussed document components from the answer set. Several types of filters have been developed in INEX, but two main types of strategies have been proposed for overlap removal: a simple method that keeps the highest ranked element of each path and more complex algorithms that take into account the relations in the tree hierarchy between the highly ranked elements. These more advanced techniques exploit the recursive structure of an XML tree.

As we shall see in Chapter 8, almost all models from Chapter 5 perform well at INEX. However, all models and in particular the XML vector space one and the language modelling ones, perform better for the thorough retrieval task than for the task aiming at returning the most focussed elements, i.e. the focussed retrieval task. We now provide a theoretical explanation for this behaviour by investigating specificity aboutness realized through XML retrieval *filters* in more detail. For this purpose, we first present an addition to our theoretical methodology in Section 7.2, which allows us to describe the reasoning of two INEX filters in Section 7.3.

7.2 Defining Specificity Aboutness

In this section, we develop our theoretical methodology to evaluate filters. We rely on some initial work by Huibers on the relationship between the filter aboutness system (characterizing the focussed task) and the corresponding underlying aboutness system (characterizing the thorough task) [Huibers, 1996], which we adapt to the requirements of XML retrieval. We go beyond his work by actually applying his theoretical work to analyse filters developed at INEX in Section 7.3.

As already explained, the task of finding the most focussed elements consists of filtering the ranked result list produced by an XML retrieval model like the ones described in Chapter 5. Generating this ranked result list is itself based on an aboutness decision system, which characterizes the model used to deliver that list. Thus, with filtering, a further aboutness decision is applied, one which removes overlapping elements from the result list.[1]

[Huibers, 1996] argues that one aboutness decision system is a filter to another aboutness decision system if the two corresponding aboutness systems are embedded — meaning their reasoning behaviour is related by supporting the same or sufficiently similar properties. In the context of XML retrieval, this translates to having to relate the aboutness decision system associated with the model for the focussed task to that of the underlying aboutness system associated with the model used to generate the ranked list (the thorough task) to then be filtered.

Our theoretical analysis of filter is done in three steps. We first formalise the translation process, as we have in Section 4.1 for retrieval models. Secondly, we identify the reasoning rules associated with the filter. Finally, we analyse the relationship between the filter and the underlying aboutness systems. For the later, we make use of the filtering function f-*answer* defined in [Huibers, 1996], which we adapt to XML retrieval:

Definition 7 *Let A_p and B_p be aboutness systems and $\mathcal{D}$ be a set of documents and Q be a query. The filtering function f-answer of A_p with respect to B_p is defined by: f-answer$(A_p; B_p; Q; \mathcal{D})$ = answer$(A_p; Q;$ answer$(B_p; Q; \mathcal{D}))$, where answer describes a function that delivers an answer set from the set $\mathcal{D}$ based on query Q.*

Using this definition, we can investigate the filtering process by looking at the relationship between f-*answer* and *answer*. [Huibers, 1996] has identified three important distinctions between f-*answer* and *answer*:

- A filtering function f-*answer*$(A_p; B_p; Q; \mathcal{D})$ is called *useless* if for all sets of documents $\mathcal{D}$ and queries Q f-*answer*$(A_p; B_p; Q; \mathcal{D}) =$ *answer*$(B_p; Q; \mathcal{D})$. An example of a useless filter is the application of the coordinate retrieval model as a filter to an answer set generated by simple vector space retrieval [Huibers, 1996], as both are based on the same aboutness decisions, according to which a document D is about a query Q if both share information items.

- Two aboutness systems preclude each other if f-*answer*$(A_p; B_p; Q; \mathcal{D}) = \emptyset$.

- The aboutness systems A_p and B_p are said to be *f-equivalent* if and only if f-*answer*$(A_p; B_p; Q; \mathcal{D}) =$ *answer*$(A_p; Q; \mathcal{D})$. An example of an *f-equivalent* filter is to use pure type XML retrieval to filter a result set generated by vector space retrieval from Section 5.2. Pure type XML retrieval defines that a document D is about a query

[1]It should be pointed out that the use of filters is not exclusive to XML retrieval. Filters are used in IR to improve performance [Huibers, 1996], if, for instance, at first a fast but less accurate approach is used to identify relevant documents from a very large set documents, and then a second retrieval system is used to search the initial result set more accurately. Pseudo-relevance feedback and passage retrieval are examples of such processes.

Q if and only if the information items of Q are a subset of the information items in D. This delivers a subset of the answer set from simple vector space retrieval, for which D is about Q if they share information. Pure type XML retrieval therefore fully determines the final answer set.

- A_p and B_p are said to *intersect* if and only if the filter is neither useless nor f-*equivalent*.

In our analysis of the relationship between f-*answer* and *answer*, we first determine whether a filter is 'useless', i.e. the filtering function does not change the original answer set. If this is not the case, next we investigate whether the filter f-*answer* uses f-*equivalent* aboutness systems. We call a filter aboutness system to be f-*equivalent*, if its A_p alone determines the final result set. If the filter is neither useless nor f-*equivalent* with regard to the underlying aboutness system, we then define how the filter and underlying aboutness system 'intersect' by comparing their aboutness properties.

We therefore slightly change our general methodology to investigate aboutness systems. Firstly, we introduce an additional step, that analyses the relationship of $f - answer$ and $answer$. Secondly, we can leave out the reflection and completeness steps, as we are only interested in estimating the impact of the reasoning behaviour of the filter on the underlying system's reasoning behaviour. In this sense, we do not consider filter presented in XML retrieval to be fully independent aboutness systems but always dependent on an underlying system. This makes them different from the filters Huibers has analysed.

The following section will demonstrate the presented methodology for the analysis of three filters implemented at INEX.

7.3 Applying Specificity Aboutness in INEX

Two main types of filters have been proposed for the focussed task at INEX: a simple model that keeps the highest ranked element of each XML path and a more complex model that takes into account the relations in the tree hierarchy between retrieved elements. For the latter, we distinguish two different approaches. The first one uses an utility prior to not simply filter out those that have been considered least relevant but to filter those which are least useful according to the prior. The second one uses a re-ranking approach to discriminate whether children elements have highly relevant parents or not.

7.3.1 Brute-force Filter

Our first method of removing overlap in the result set of an XML retrieval model has also been referred to as 'brute-force filter', because only the highest ranked element from each of the XML paths is selected. The advantage of this filter is that it is relatively easy to implement and that it can be used on top of any kind of underlying aboutness system. It is independent of the specification of the underlying aboutness relation. The disadvantage is of course that it is not always the case that the highest ranked element in a path is also the most useful one [Mihajlovic et al., 2005].

7.3.1.1 Aboutness Decision

Given a document d and a query q as well as set of descriptors $\chi(d)$ and $\chi(q)$:

$$d \text{ about } q \text{ if and only if } rsv(\chi(d), \chi(q)) = max(rsv_u(\chi(d), \chi(q)))$$

$max(rsv_u(\chi(d), \chi(q)))$ is delivering the XML element with the maximum retrieval status value for the underlying aboutness system. For the translation, let d be a document component, e_n element types, k_n values in an element and i an identifier to enumerate all $\{1, ..., n\}$ elements in an XML tree in a depth-first traversal manner: $map(\chi(d)) = \{\langle\langle ElementType, e_1, i_1\rangle\rangle, \langle\langle ElementType, e_2, i_2\rangle\rangle, \langle\langle Parent, i_1, i_2\rangle\rangle, ..., \langle\langle ElementType, e_n, i_n\rangle\rangle, \langle\langle Parent, i_{n-1}, i_n\rangle\rangle, \langle\langle Value, e_n, k_1\rangle\rangle, ..., \langle\langle Value, e_n, k_n\rangle\rangle\} | \forall i_i \in \{\langle\langle Parent, i_i, i_k\rangle\rangle\}, count(i_i) = 1\}$.
The translation expresses that we only consider elements on the same XPath, meaning each element is the parent and the child of exactly one other element, unless it is the root or leaf element.

If $max(rsv(\chi(d), \chi(q)))$ is the maximum retrieval status value in any relevance decision on an XML path, then for the Gardens Point model, e.g., this is $max(D(c)\sum_{i=1}^{c} rsv_{L_i})$, where $L_i = K^{n-1}\sum_{i=1}^{n}\frac{t_i}{f_i}$.

7.3.1.2 Reasoning Behaviour

We now continue analysing the functional behaviour of brute-force filtering using the reasoning rules from Section 4.4. Regarding the reasoning behaviour, filters are different from what we have seen before. We need to argue about them in relation to the underlying answer system $answer(A_p; Q; D)$ and see whether they produce contradictions. Please note that for all filters neither containment nor preclusion are defined, because filters do not consider one piece of information to be contained in another piece of information nor do they analyse information clashes. Thus, we ignore all the rules related to them.

Reflexivity holds for brute-force filtering. A maximum element will be about itself. **Set Equivalence** is also given. We only prove Left Set Equivalence. Assuming that $S \equiv T$ and $S \Box\!\!\leadsto U$ hold, we have to show that $T \Box\!\!\leadsto U$ is given. From the premises, we know that S is the highest ranked document component situation on the path for U. As S and T are equivalent, T will have the highest rank, too. Set Equivalence will be supported.

For **Symmetry**, there is no contradiction in the statement that if $S \Box\!\!\leadsto T$ or S is the highest ranked answer to T then also $T \Box\!\!\leadsto S$ or T will be the highest ranked answer to S. It is possible that the same aboutness system allows this. Brute-force filtering is symmetric. Let us use the example of Symmetry to look at how the second aboutness system of filters influences underlying aboutness systems. As Symmetry is fully supported, it does not change the underlying aboutness behaviour. All systems, for instance, that are based on overlap aboutness, are symmetric. Examples include the XML vector space retrieval model and XML language modelling presented in Chapter 5. All these systems remain symmetric if brute-force filtering is applied on top of them. Underlying non-symmetric aboutness systems remain non-symmetric. This means that those reasoning properties that are fully supported by a filter do not change the underlying aboutness behaviour of the system that produced the original ranking. If, however, a reasoning property such as Symmetry is not fully supported by a filter or, for instance, a threshold is applied, then this naturally changes the underlying aboutness behaviour, as we shall see with Transitivity next.

Most interesting are those reasoning rules that are not supported. These definitely change the underlying aboutness behaviour. The **Transitivity** rule, for instance, is not supported for brute-force filtering, as two situations cannot be the highest ranked answers towards the same query. If T is the highest ranked answer to U, S cannot be the highest ranked answer to U, too. This means whatever the status of Transitivity in an aboutness system, if we apply brute-force filtering on top of it, it will not be supported. An example of a fundamentally changed model would be pure type XML retrieval, for which Transitivity was fully supported. The same applies for **Euclid**: If S is the highest ranked answer to U, how could T be the highest ranked answer to the same U, too? This means Euclid is never supported.

Left Monotonic Union (LMU) would imply in the context of brute-force filtering that if one extends S to $S \otimes U$ and aboutness would be preserved for both, both S and $S \otimes U$ would be highest ranked answers, which is a contradiction. This means LMU is not supported either. That LMU is not supported, is a change of the aboutness behaviour of all XML retrieval systems from Chapter 5. **Right Monotonic Union** (RMU), however, is given and the underlying aboutness system's behaviour is not altered. S can be the highest ranked answer to both T and $T \otimes U$. This is, for instance, the case for any aboutness system where U does not contribute to the aboutness. We have not seen such a case in Chapter 5, but it is at least theoretically possible.

Mix is another rule that cannot be supported, with again a strong impact on the reasoning behaviour of many models from Chapter 5, which support it. It states that with the assumptions $S \Box\!\!\rightsquigarrow U$ and $T \Box\!\!\rightsquigarrow U$, we can also say that $S \otimes T \Box\!\!\rightsquigarrow U$. S and T, however, cannot be at the same time the maximum answer to U, unless S and T are on different XML paths. Then, however, it is a contradiction that both S and $S \otimes T$ are maximum scores to U. Mix is not supported.

Regarding **Cut**, from $S \otimes U \Box\!\!\rightsquigarrow T$ and $S \Box\!\!\rightsquigarrow U$, we can only conditionally conclude that $S \Box\!\!\rightsquigarrow T$. If $S \otimes U$ is the highest ranked answer to T, then only S and S alone can be the maximum answer. This means $S \Box\!\!\rightsquigarrow T$ if and if only $S \otimes U \equiv S$. This would be the case as Left Set Equivalence holds for brute-force filtering. This would be, however, a very special and somewhat degenerated version of Cut. Thus, we do not consider Cut to be supported. Similar arguments apply to **Right Weakening**. With $S \Box\!\!\rightsquigarrow T \otimes U$, we are able to conclude $S \Box\!\!\rightsquigarrow T$ if U does not contribute to the fact that S is the maximum retrieval status value to the topic $T \otimes U$. For Gardens Point (Section 5.4.1), for instance, using brute-force filtering, this is the case if, e.g., none of the information needs in U is answered by S so that for all terms in C (with $U \equiv map(C)$) $t_i = 0$ in $rsv_L = K^{n-1} \sum_{i=1}^{n} \frac{t_i}{f_i}$. As this is again only an extreme case, we consider Right

Weakening to be not given.

Regarding **Context-Free And**, with $S \Box\!\!\rightsquigarrow T$ and $S \Box\!\!\rightsquigarrow U$, we could also say that $S \Box\!\!\rightsquigarrow T \otimes U$. Context-Free And is fully supported, as it is an extension to Right Monotonic Union. So the underlying behaviour is not changed. We assume that S is the highest scoring answer to T and U. Then, combining $T \otimes U$ does not change that. It either increases or does not change the overall relevance of S. Context-Free And is fully supported.

All the rules analyzed in this section are important for the analysis of XML retrieval models' behaviour. When we analyze the experimental results related to brute-force filtering in Section 8.4, we shall see the impact of excluding the rules' reasoning behaviour. We continue with analyzing $f - answer$.

7.3.1.3 F-answer

In this section, we look at the relation between f-*answer* and *answer*. First, we need to show that the brute-force filter is not *useless*. This can be formally proven by demonstrating that the aboutness systems of a filter and its underlying system differ in at least one reasoning characteristics — be it a certain rule, be it a single condition of this rule. We have just seen that brute-force filtering disallows LMU, Transitivity, etc., which means it is not useless as a filter for XML retrieval models from Chapter 5. As $max(rsv_u(D, Q))$ is dependent on the underlying retrieval status value rsv_u, brute-force filtering is also not f-*equivalent*.

As the filter is neither useless nor f-*equivalent*, neither brute-force filtering nor the underlying aboutness systems from Chapter 5 fully determine the outcome of combining both. They 'intersect', and we need to look at the differences in reasoning behaviour, the filter creates: E.g., LMU reasoning is excluded, which changes any aboutness system that follows the strict structural constraints of XML documents: If an element is a child, it shares information with its parent. This means for language modelling from Section 5.3, for instance, that both are about the same queries. Yet, such aboutness due to overlap in information is what is supposed to be excluded by brute-force filtering. It just does not discriminate whether the overlap is due to new relevant information or due to redundant information.

We continue with our investigations with models, which argue that brute-force filtering is not fine-grained enough. First, we analyse how a utility prior shall improve the outcome of brute-force filtering before finally we look at an alternative approach based on re-ranking.

7.3.2 Utility Prior

In [Mihajlovic et al., 2005], again brute-force filtering is used to remove all overlapping elements in a path. However, the authors argue that this cannot be done by simply selecting the highest ranked element on a path, and one needs to consider the 'usefulness' of an element compared to other elements on the same path. They suggest to measure the 'usefulness' with a function $U(d)$. This function is equivalent to giving an utility prior to the retrieval status values by considering the original relevance, the element's size and its irrelevant information. Smaller elements, e.g., potentially have less useful information.

The utility prior of any element is estimated by [Mihajlovic et al., 2005] as follows:

$$U(d) = (1 - \frac{\sum_{i \in nrch(d)} size(i)}{size(d)}) * rsv(d,q) * size(d)$$

$rsv(d,q)$ is the original relevance. $nrch(d)$ is the set of non-relevant children of d, for which $P(d) * size(d)$ is lower than a quality threshold. The authors used several standard retrieval models in their experiments.[1]

$U(d)$ can be rewritten as $U(d) = (size(d) - \sum_{i \in nrch(d)} size(i)) * rsv(d,q)$. This shows clearly how the original relevance score is influenced by the prior. The more non-relevant children there are and the larger their size, the smaller the resulting retrieval status value after applying the filter. All thresholded aboutness decisions ($rsv(d,q) > \theta$) are changed. As seen in Chapter 5, examples include the XML vector space model and the language modelling approaches.

7.3.2.1 Aboutness Decision

As we are considering a prior to brute-force filtering, the brute-force aboutness decision is not changed with the exception of the inclusion of the prior. The calculation of the final retrieval status value is then based on the prior influenced relevance score of d. Thus, d about q if and only if $rsv(\chi(d), \chi(q)) = max(rsv(U(\chi(d)), \chi(q)))$, in order to keep just the highest ranked elements in a path. We do therefore not investigate how the utility prior is a new filter aboutness decision, but how it influences the brute-force one. As the final aboutness decision is still the one for brute-force filtering, we only need to investigate those reasoning properties that hold for it.

7.3.2.2 Reasoning Behaviour

Reflexivity is fully supported by brute-force filtering. With the prior, it is conditionally supported, as the prior is only applied to the document component but not to the query. Then, it is possible that though a document component would be about itself that, with the prior, a document component would be not about itself, as $\chi(d) \not\equiv U(\chi(d))$. This is the case, if, for instance, the prior lowers the aboutness relation below its threshold. Only under the condition that the element is useful, it is also about itself.

Transitivity and **Euclid** did not hold for brute-force filtering. **Set Equivalence** is given. We only prove Left Set Equivalence. If the assumptions of $S \equiv T$ and $S \Box\!\!\leadsto U$ hold, then also $T \Box\!\!\leadsto U$. If S and T are equivalent, they have the same non-relevant children with the same size. Thus, $U(A)$=$U(B)$ with $S \equiv map(A)$ and $T \equiv map(B)$. As S and T are equivalent, Set Equivalence is supported.

[1]Please note, that we were not able to find an exact definition for *size* in [Mihajlovic et al., 2005], so that we assume that it is implemented with some standard such as counting the number of information items in the element.

Symmetry does not hold for the utility prior. We can easily give an example so that $S \Box\!\!\rightsquigarrow T$ but not $T \Box\!\!\rightsquigarrow S$. Let us, for instance, assume that we have an aboutness system based on information overlap such as the LM I system from Section 5.3.3.4. Let us furthermore assume that $S \equiv map(A)$ and $T \equiv map(B)$. Then, though $rsv(U(A), B) > 0$, it can still be the case that $rsv(U(B), A) \leq 0$.

Left Monotonic Union (LMU) does not hold for brute-force filtering, while **Right Monotonic Union** (RMU) does. With RMU, we would be able to say that $S \Box\!\!\rightsquigarrow T \otimes U$ given that $S \Box\!\!\rightsquigarrow T$. Say, $S \equiv map(A)$, $T \equiv map(B)$ and $S \otimes U \equiv map(C)$. How does RMU change with the utility prior? Let us as an example study thresholded overlap decisions such as in the Contextualisation model from Section 5.4.2. With respect to $U(d) = (size(d) - \sum_{i \in nrch(d)} size(i)) * rsv(d, q)$, the newly added information can have a lot of non-relevant children with large sizes and with high impact on the Contextualisation retrieval method. We have investigated in Section 5.4.2.3 how the seemingly simple extension to include negative weights has a strong impact on the monotonicity behaviour. Similar arguments apply here.

We have to discriminate two cases: (1) The newly added information is content added to D. Then, even if this content is irrelevant or highly undesired, the size of the element will increase while no changes to the number of non-relevant children happen. This means that $(size(d) - \sum_{i \in nrch(d)} size(i))$ increases for any kind of newly added content. This might be counter-productive if we would like to control the monotonic behaviour with a threshold as in the Contextualisation method. We could imagine the case where the added content lowers the overall retrieval status value below the threshold but the utility prior raises it again above the threshold. This is the case, as the utility prior is not dependent on content changes apart from the size measure, which for newly added information has to increase.

Now let us take the second case: (2) We add new document components and not just content. As seen in Section 5.4.2, with Contextualisation an element that is about a query can be put into a context of other document components with many non-relevant children or large sizes. The threshold could be missed. Here, the monotonic behaviour is controlled by adding undesired document components, but not by adding undesired content. Otherwise, similar arguments apply as for case 1.

If RMU is fully supported by an XML retrieval model, the utility prior might change this behaviour by introducing a new level of control. This explains why it worked better in the experiments with language modelling than with Gardens Point [Mihajlovic et al., 2005]. Gardens Point does not support RMU, so no changes apply. The language models in Section 5.3, on the other hand, fully support RMU, and now have an additional means of controlling the monotonic behaviour better.

Cut and **Right Weakening** are only under a very particular condition part of the aboutness model for brute-force filtering so that we can ignore them here. **Mix** has not been given for brute-force filtering. Regarding **Context-Free And**, with $S \Box\!\!\rightsquigarrow T$ and $S \Box\!\!\rightsquigarrow U$, we could also say that $S \Box\!\!\rightsquigarrow T \otimes U$. Context-Free And is an extension to Right Monotonic Union, so we suspect that it is at least conditionally given. Here, however, we still have the second assumption that $S \Box\!\!\rightsquigarrow U$, which makes it impossible that the newly added information in U might lead to a threshold in the underlying aboutness decision to be missed. Context-Free And is therefore fully given and the underlying aboutness behaviour not changed.

7.3.2.3 F-answer

The utility prior is based on brute-force filtering. This means it inherits most of the underlying filter characteristics of it. It is not useless, as our discussions of thresholded retrieval models show. It changes most of the thresholds. It also forces systems to be non-symmetric. However, not in all cases it also changes the aboutness behaviour. For instance, it still fully supports Set Equivalence, which is also supported by all models from Chapter 5. This means it is not necessarily contradicting the underlying aboutness decision. The utility prior is not f-*equivalent* either, as it depends on brute-force filtering. Thus, it is 'intersecting' with the behaviour of the underlying aboutness system. In the discussion of the reasoning behaviour, we have seen that the main influence of the utility prior stems from its impact on thresholds or from introducing thresholds.

We do not need to discuss again the impact of the brute-force filter on the changes in reasoning behaviour. We can focus on the impact of the utility prior. Its main impact is that it 'conditionalises' reasoning properties that would otherwise be fully supported. This is particularly noticeable for Reflexivity. An element is not just about itself anymore. It has to be useful, too, which not only depends on its relevance but also on its size, as these are the two main factors to influence its usefulness. This is a clear change to any of the considered aboutness systems for which Reflexivity was given. It is only the Contextualisation model that does not support Reflexivity. Thus, the prior has no influence on it at this point. But generally speaking, all models are changed if Reflexivity or its sister rule Singleton Reflexivity are influenced by the prior.

Most of the prior's impact on the monotonic behaviour has already been discussed so that we do not need to repeat it here. Let us additionally look at the impact on a threshold of one of the models from Chapter 5. For LM I (Section 5.3.3.3), the threshold θ is an internal one or based on the smoothing value, which depends on the collection language model $P_{mle}(t_i)$. θ does now change. It becomes dependent on the size of the non-relevant document components, too.

Furthermore, Right Monotonic Union is only conditionally given for the prior. This, e.g., influences the aboutness decision of the LM II language model (Section 5.3.4). The internal condition θ there is 'externalised' with the prior being an additional external condition.

[Mihajlovic et al., 2005] conclude that their approach to re-rank retrieval scores using an utility function seems to improve effectiveness when removing overlap, but does not outperform the simple approach of selecting the paragraph elements. This might be the case, as their approach is not actually based on the structure of a document, but on the information overlap between different components. The structure is therefore only indirectly exploited with similar consequences as discussed for various models from Chapter

5.

The next filtering approach considers overlap as something that is not always to be avoided.

7.3.3 Controlling the Overlap: Re-ranking

The next approach [Clarke, 2005] we are going to present re-ranks the elements with a new context-dependent retrieval status value and does not entirely eliminate overlapping elements. It is iterating through the following three steps to control overlap:

1. Report the highest ranking element.
2. Adjust the retrieval status values of the unreported elements.
3. Repeat steps 1 and 2 until m elements are reported.

Overlap is controlled by reducing the relevance of elements parenting highly ranked elements in step 2, as they might be highly relevant only because they have so many relevant children. Controlling overlap like this is a more conservative approach than radically eliminating it by just allowing the highest ranking element in any one path. For this reason, we have chosen to discuss this approach here. [Clarke, 2005] rightly argues that some overlap might be beneficial.

In [Clarke, 2005], the input into the re-ranking method is a list of n elements each associated with $x.\overrightarrow{f}$ as the term frequency vector per query term, with $x.\overrightarrow{g}$ as an adjustment vector, with $x.l$ as the element length, with its current score $x.score$ as well as other information required to process the algorithm such as the set of children per node. Please note that the *score* field in [Clarke, 2005] is based on $BM25$, hence the adjustment of $x_t = f_t - \alpha * g_t$ in the *score* function: $\sum_{t \in Q} w * q_t \frac{(k_1+1)x_t}{K+x_t}$.

We focus on the intention of the algorithm to lower the weights of those document components containing highly relevant children. The adjustment of x_t to $x_t = f_t - \alpha * g_t$ works in the presented algorithm in two ways. For parents y containing a highly relevant child x their adjustment score $y.\overrightarrow{g}$ is increased for those terms previously reported: $y.\overrightarrow{g} = y.\overrightarrow{g} + x.\overrightarrow{f} - x.\overrightarrow{g}$. With $x_t = f_t - \alpha * g_t$, this means in effect that the overall retrieval status value of such parents is reduced for already recorded terms depending on α. For the children of highly ranked parents, we know that its terms have already been considered in the reported parent element. Hence, its $x.\overrightarrow{g}$ becomes $y.\overrightarrow{f}$. The retrieval status value is recomputed with $x_t = f_t - \alpha * g_t$, and the impact of reported terms reduced by $(1 - \alpha)$ as $x_t = f_t - \alpha * f_t$.

The overall algorithm has as an input a priority queue S, containing XML elements ranked by their initial scores, and returns its results in a priority queue F, which contains the top m ranked elements. There are two tree traversal routines: *Up* and *Down*. The *Up* routine removes each ancestor node from S, adjusts its term frequency values, recomputes its retrieval status value and adds it back into S. The adjustment of the term frequency values adds only the previously unreported term occurrences. The *Down* routine performs a similar operation on each descendant. However, since the contents of each descendant are entirely contained in a reported element, its final score is computed. Finally, the element is removed from S and added to F.

7.3.3.1 Aboutness Decision

Only m top-ranked elements are considered to be included in S. This might lead us to the assumption that we need to consider a thresholded aboutness decision. However, according to the algorithm, no element is filtered out of the result list F unless the adjusted score becomes 0. Therefore:

$$d \text{ about } q \text{ if and only if } rsv_{adjusted}(\chi(d), \chi(q)) > 0$$

$rsv_{adjusted}$ is based on the algorithm just described. $\chi(d)$ and $\chi(q)$ are again the set of descriptors for document component d and query q. rsv is particularly dependent on α, as the discussions below show. The adjusted score becomes 0 if for all existing query terms their frequency is 0. This applies to all underlying aboutness decisions we have covered this far, as they all use the sum of all query terms scores to determine the overall score. This means for the $rsv_{adjusted}$ that $f_t = \alpha * g_t$.

Structure comes into play when considering the next element to look at by moving up and down in the tree. The model is based on a full tree traversal and registers children and parents of elements, but does not consider attributes. The translation function is therefore the same as for pure type XML retrieval from Section 4.7.

7.3.3.2 Reasoning Behaviour

The first reasoning property to demonstrate is **Reflexivity**, which states that $S \Box\!\!\leadsto S$. Reflexivity is not given. With $S \Box\!\!\leadsto S$, then $f_t = g_t$. If in $x_t = f_t - \alpha * g_t$, and $\alpha = 1$ according to [Clarke, 2005], then $x_t = f_t - 1 * g_t$, which means $rsv_{adjusted} = 0$, with $f_t = g_t$. Thus, Reflexivity is not supported.

However, Reflexivity is a special case and the exception. Generally speaking, re-ranking does not fundamentally change the aboutness decision of the XML retrieval models but adds emphasis to the ranking of elements. For our analysis of the impact of filters we therefore need to relate it directly to the models we have developed in Chapter 5. Re-ranking's main effect is on thresholded aboutness decisions, which we have not met for many aboutness rules but mainly for rules related to monotonic reasoning. Two examples of rules that were either fully or not at all supported are **Symmetry** and **Transitivity**. Re-ranking has no influence on both reasoning behaviours.

We want to therefore concentrate on **Left Monotonic Union** (LMU), as it is a reasoning property in Chapter 5 controlled by thresholds. LMU would be given if $S \otimes U \Box\!\!\leadsto T$ and $S \Box\!\!\leadsto T$ are given. For the XML vector space model from Section 5.2, re-ranking with $x_t = f_t - \alpha * g_t$ can of course reduce the extension to fall below the threshold n. This mainly affects the children of the highly ranked parents. LMU is only conditionally supported if re-ranking does not lower the retrieval result to fall below n. An interesting case forms the language modelling approach in Section 5.3.3.3. Its internal threshold based on the smoothing value might be missed if the added information leads to a re-ranking with a lower retrieval status value. Therefore, applying re-ranking on top of LM I means that LMU is now conditionally supported, while LM I alone fully supports LMU. Similar arguments apply to **Right Monotonic Union**.

Next, we consider **Mix** and **Context-Free And** as derivatives of LMU and RMU, respectively. Mix adds to LMU that $S \Box\!\!\rightsquigarrow U$ and $S \Box\!\!\rightsquigarrow T$ and therefore $S \Box\!\!\rightsquigarrow T \otimes U$. It is fully supported by re-ranking as the additional assumption that $S \Box\!\!\rightsquigarrow U$ take cares that $S \Box\!\!\rightsquigarrow T \otimes U$ is not an element to reduce below a threshold after adjusting with $x_t = f_t - \alpha * g_t$. Regarding Context-Free And, with $S \Box\!\!\rightsquigarrow T$ and $S \Box\!\!\rightsquigarrow U$, we could also say that $S \Box\!\!\rightsquigarrow T \otimes U$. Just like Mix, it is fully given.

7.3.3.3 F-answer

Re-ranking is certainly not *useless*, because the LMU thresholds for vector space retrieval and language modelling, for instance, have been changed. It is not f-*equivalent* either, as it is dependent on the underlying aboutness decision, because re-ranking is a function of the original retrieval status value. Thus, re-ranking is also 'intersecting'. Reflexivity is changed through the impact of α. That Mix behaviour is preserved is a clear advantage towards the brute-force filtering approach, as it is an important property of XML retrieval. The support for Mix adds to the better performance in the experimental results, which we shall look at in Section 8.4.

Looking at re-ranking, it is difficult to make general statements regarding its impact on XML retrieval, as it has been developed for a particular model. The authors of the re-ranking approach, however, report limitations of their algorithm according to their experimental evaluation [Clarke, 2005]. From a theoretical evaluation point of view, an immediate recommendation on how to potentially improve the ranking would be to introduce a threshold to control the monotonic behaviour of the re-ranking aboutness decision: Only if $rsv_{adjusted}(\chi(d), \chi(q)) > \theta$, the element would be reported. We have seen in Chapter 5, that thresholds effectively add to the control of the monotonic behaviour and improve performance. The advantage would be that also those reasoning properties, that are either fully or not at all given, could be influenced through re-ranking, if a threshold were introduced.

We have considered filters as a second layer aboutness decision and asked whether they are able to influence the underlying aboutness system for the better. We could do so, as we regarded them as aboutness decisions in their own right and looked at how they change the specificity context.

7.4 Conclusion

In this chapter, we have asked how filters can be suitable extensions to the underlying aboutness decision so that just most relevant elements are returned. We could show the impact of the brute-force filter. It almost completely changes, e.g., the monotonic behaviour. The XML vector space retrieval model, for instance, has been overall very successful in the experimental evaluation in INEX [Mass and Mandelbrod, 2005], but its performance decreases for the tasks to deliver only non-overlapping document components, ranked according to how specific they are to the query. The situation is similar for the XML retrieval model based on language modelling [Sigurbjörnsson and Kamps, 2005]. Its performance decreases, too, when brute-force filtering is used to filter the original language modelling retrieval results.

Reasoning behaviour	*Brute-force filtering*	*Prior*	*Re-ranking*
Reflexivity	fully	not	not
Symmetry	fully	not	N/A
Set Equivalence	fully	fully	N/A
Transitivity	not	not	N/A
Euclid	not	not	N/A
LMU	not	not	$x_t = f_t - \alpha * g_t$
RMU	fully	$U(D)$	$x_t = f_t - \alpha * g_t$
Cut	not	not	N/A
Right Weakening	not	not	N/A
Mix	not	not	fully
Context-Free And	fully	fully	fully

Table 7.1: Filter aboutness

If we try to understand why brute-force filtering decreases performance in XML retrieval, two changes in reasoning properties are highly conclusive:

1. LMU is not supported by brute-force filtering. The XML vector space retrieval model, for instance, successfully uses conditions on LMU reasoning to adjust the behaviour of flat document vector space retrieval to the requirements of XML retrieval. This ability is lost once the brute-force filter is applied, which explains a decrease in performance.

2. Mix is not supported by brute-force filtering. Among other things, Mix describes that, if two children D and D' are about a query, then their parent item $D \otimes D'$ will also be about the same query. This behaviour is typical of XML based resasoning. If it is not supported, problems might arise, such as the elimination of potentially highly relevant children. We shall discuss this point in more detail in Chapter 8 when we discuss the interaction of brute-force filtering with all the models from Chapter 5 and its impact on experimental results in INEX.

The utility prior might change the monotonic behaviour by introducing a new level of control if Left Monotonic Union is fully supported by a model. It works more effectively with models that originally fully supported Left Monotonic Union. Lastly, the re-ranking method did not offer the expected impact on monotonic behaviour in that it added a new level of control to the reasoning.

Table 7.1 summarizes the results of our theoretical evaluation for XML retrieval filters. As discussed, for filters to support a reasoning property means not to change their underlying reasoning behaviour. If filters conditionally support a reasoning property, they will add a condition to it or change an existing one. If they do not support a reasoning property, they will eliminate it from the overall aboutness behaviour. In Table 7.1, we can clearly identify the strong impact of brute-force filtering on the XML retrieval models from Chapter 5.

In the next chapter, we use all the results from Chapters 5, 6 and 7 to try to understand the experimental results obtained at INEX.

Chapter 8

Theoretical Analysis of the INEX Experimental Evaluation Results

8.1 Introduction

Chapter 3 argued that experimental and theoretical evaluations can be complementary. We would now like to specify the complementarity of experimental and theoretical evaluation by demonstrating how our theoretical evaluation results can explain experimental behaviour. We mainly look at how results from the INEX 2005 experimental evaluation can be explained using insights from our theoretical evaluation. We use INEX 2005 only in order to have a baseline and because many models have not changed fundamentally since then.

As the work would otherwise be too extensive, we concentrate on the INEX 2005 ad-hoc content-only (CO) tasks and on the comparison of the XML vector space model with the XML language model, the Gardens Point model and the Contextualisation method, all of which were extremely successful in the INEX 2005 campaign and consistently ranked among the top models [Fuhr et al., 2006]. Concentrating on successful models helps us demonstrate the ability of a theoretical evaluation to analyse also minor differences in performance.

In this chapter, we proceed as follows: Firstly, we introduce all strategies in Section 8.2. Secondly, we go through each of them one by one and look at the results of the models from Chapter 5. Finally, we draw conclusions on their individual performance also with respect to their relative performance compared to other models.

8.2 Retrieval Strategies

The main retrieval task in INEX 2005 was the ad-hoc retrieval of XML elements [Kazai and Lalmas, 2005]. This task involves searching a given amount of documents based on a varying sets of topics. In INEX, XML elements may be retrieved instead of documents. For INEX 2005, several retrieval strategies were defined for ad-hoc subtasks based on CO queries:

1. Thorough: In this strategy the problem of overlap is to be ignored by models. Overlapping elements are supposed to be mainly a presentation issue, and an interface would offer the user various ways of browsing through a set of relevant but potentially overlapping elements. Due to the hierarchical inclusion of elements, large number of overlapping elements are returned. This task is a challenge for models, as their ranking is supposed to return highly exhaustive and specific elements first.

2. Focussed: According to this evaluation strategy, retrieval models are rewarded most for the best return of focussed XML elements, i.e. those at the right level of granularity. Overlapping elements should not be returned. This means that the most exhaustive and specific element on a path is to be delivered by the model. Generally speaking, in the task preference is given to specificity.

3. FetchBrowse: This strategy investigates achieving the best mixture of document retrieval and element retrieval strategies. In the fetching phase, relevant articles are

identified in order to return the most exhaustive and specific elements from within these relevant articles in the browsing phase.[1]

We now continue with suggestions on how to explain experimental test results using our insights from the theoretical evaluation in Chapters 5, 6 and 7. In the following sections, we go through several snapshots of performances for the retrieval models; first in the Thorough strategy, then in the Focussed strategy and finally in the FetchBrowse one. The order is important here, as Focussed is often a post-processing of the initial retrieval in Thorough using filters, while FetchBrowse is a mixture of the first two evaluation strategies.

For Thorough, we are mainly interested in finding some important reasoning rules that help explain experimental behaviour, while for Focussed we are interested in how reasoning rules interact with filters from Chapter 7. Here, the most important filter is the brute-force one from Section 7.3.1, for which we have already proven that it intersects with underlying reasoning behaviour in Section 7.3.1.3. Each performance snapshot is then represented in Tables 8.1 to 8.7, which each contain a relative comparison of Gardens Point, XML vector space, XML language modelling and Contextualisation method for various tasks in INEX 2005. We are going to use our theoretical evaluation results to explain the absolute and relative performance of models within these snapshots of INEX 2005 performance results and start with INEX 2005 Thorough.

8.3 Thorough

For the Thorough task in INEX 2005, we concentrate on the metrics effort-precision/gain-recall ($ep - gr$). Effort-precision [Kazai and Lalmas, 2005] is based on the idea to measure the amount of relative effort that a user has to make to find the right information in the real ranking compared to the effort an ideal ranking would take. We proceed as follows. First we introduce some of the background of the metrics for the thorough task. Next, we analyse which of the aboutness reasoning properties influence the results under these particular metrics. Finally, we investigate the way in which models from Chapter 5 implement these reasoning properties helps explain their experimental performance for effort-precision.

Effort-precision ep is defined in [Kazai and Lalmas, 2005] by $ep[r] = \frac{i_{ideal}}{i_{submission}}$. i_{ideal} is measured as the rank position at which the cumulated gain of r is reached by the ideal curve of the ranking. $i_{submission}$ is measured by the same rank position in the real submission.

i_{ideal} and $i_{submission}$ are best explained with an example. Let us assume that we have a ranking of three elements i_1, i_2 and i_3. i_1 has the retrieval status value 2, i_2 1 and i_3 0. A real model returns $\{i_3, i_1, i_2\}$. In an ideal model we had the following ranking: $\{i_1, i_2, i_3\}$. The cumulated gain of 2 would be reached by the ideal model at rank 1, while the real model delivers it only at rank 3. This means $ep[r] = 1/3$.

ep is captured at gain-recall points gr [Kazai and Lalmas, 2005], where gain-recall is calculated as the cumulated gain value divided by the total achievable cumulated gain.

[1]During INEX 2006, the FetchBrowse task was renamed to 'Relevant in Context'.

The similarity to traditional precision/recall evaluation models is obvious. $ep-gr$ therefore measures where the most relevant XML elements are found in the ranking produced by a model. As one would expect, the more they are concentrated in the tail of the result list, the worse the performance.

Secondly, $ep-gr$ measures how completely the set of most relevant elements is represented at the top of the ranking. As the formula stands, a small number of highly relevant elements that appear at the tail of the ranking have a strong impact on the performance. This is the case, because the cumulative gain measure penalises anything that is not in perfect order and because we divide by $i_{submission}$, which, for instance, would give an element that was the most relevant one but only appeared at position 1000 a value of 1/1000.

The formula of cumulative gain is constructed in such a way that all elements following an element that is not correctly placed according to its ideal rank are penalized. This becomes clear, if we look at an example taken from [Kazai and Lalmas, 2005]. In the example, an ideal ranking would return on top a list of elements with $\{3, 3, 2\}$ as retrieval status values. Now, let us assume two less perfect models, the first one produces an element ranking with $\{3, 2, 2\}$, while the second one produces a ranking with the retrieval status values $\{2, 3, 3\}$. For the first model the gain vector xG is $\{1, 0.86, 0.875\}$ with an average of 0.91, while it is $\{0.66, 0.86, 1\}$ for the second one with an average of 0.84. Though the second model returns more quickly all the most relevant elements, it performs worse according to ep, as it misjudges the most relevant element.

The effects of the cumulated gain behaviour is amplified by the way effort-precision ep is defined by $ep[r] = \frac{i_{ideal}}{i_{submission}}$, where r is the rank. Then, in the above two examples if we consider each element as a cut-off point for ep and use a regular growth of i_{ideal}, the first one has as $ep[r]$ $\{1, 0.83, 0.83\}$ with an average of 0.89. For the second example, it is $\{0.66, 0.83, 1\}$ with an average of 0.83, which again makes the second model perform worse than the first. Thus, $ep-gr$ punishes those models more that do not deliver the most relevant elements on top of the rankings though their overall ranking might still be good.

Such behaviour of the $ep-gr$ measure has its roots in the relationship of the series for the cumulated gains in numerator and denominator. Both series are sequences of partial sums (cumulated gains) that converge towards the maximum cumulated gain. In our example the maximum cumulated gain is 8. The denominator series of the ideal submission grows quickly and regularly for lower ranks, as here the most relevant elements are found. It converges more quickly to the maximum cumulated gain than the numerator, which measures the cumulated gain of the real submission. Differences between numerator and denominator are therefore more significant in lower ranks. Thus, (1) all elements after a wrongly placed element in the ideal submission are penalised, and (2) those models perform worst that do not deliver all most relevant element on top of their ranking.

In the next section we introduce three (example) reasoning rules and how they are relevant to performance in $ep-gr$.

Table 8.1: INEX 2005 Thorough with the $ep - gr$ metric and generalised quantisation

S.No	Affiliation	RunId	MAep
1	XML vector space	no-phrase-...	0.0867
2	XML vector space	no-phrase-...	0.0841
4	Language Model	UAmsCOTQrelbasedIndex	0.0829
5	Language Model	UAmsCOTLengthbasedIndex	0.0802
6	Language Model	UAmsCOTElementIndex	0.0793
7	Gardens Point	QUT 1-Thorough	0.0757
12	Contextualisation Method	Tampere-b01-o-root	0.0726
13	Contextualisation Method	Tampere-b003-o-tower	0.0725
14	Gardens Point	2-Thorough	0.0706
15	XML vector space	with-phrase-...	0.0698
17	Contextualisation Method	Tampere-exp10-b01-root	0.0686

8.3.1 Rules Relevant to Effort-precision/Gain-recall

We want to find the reasoning rules that help models deliver the most relevant elements as early as possible. To this end, we need to look at how models can preserve aboutness. Those elements that are about a query are related in the information they have — otherwise they would not be about the same query. Then, the question of how to deliver those elements first that are about the same query becomes a question of the ability to preserve aboutness over all the most relevant elements. Having found a relevant element, aboutness for all its related relevant elements has to be preserved for $ep - gr$.

Related relevant XML elements differ in how much relevant information they have. A relevant element that has at least the same relevant information or more can be found by either applying Left Monotonic Union and Mix reasoning. As defined in Section 4.4, LMU concludes that $D \otimes D' \Box\!\!\rightsquigarrow Q$, given $D \Box\!\!\rightsquigarrow Q$. With respect to Mix, we can conclude from the assumptions $D \Box\!\!\rightsquigarrow Q$ and $D' \Box\!\!\rightsquigarrow Q$ that also $D \Box\!\!\rightsquigarrow D' \otimes Q$, where D, D' and Q are situations. Cut reasoning, on the other hand, derives those related relevant elements that have at least the same relevant information or less. Cut states that with $D \otimes D' \Box\!\!\rightsquigarrow U$ and $D \Box\!\!\rightsquigarrow D'$, then also $D \Box\!\!\rightsquigarrow Q$.

LMU, Mix and Cut are important rules, that determine how a model decides whether those elements D that share relevant information are also about the same query. But, supporting these rules does not automatically help with a good performance under $ep - gr$. In the case of LMU, for instance, we already know from our discussions in Chapter 5, that the added information Q' does not necessarily have to be about the query. Models, which support LMU, have therefore problems returning only highly relevant elements early in the ranking. We would expect that models that are at least conservative in their (left) monotonic behaviour perform better than those which fully support LMU. Regarding Mix, the added D' is also about Q. Therefore, Mix generally supports better performance in $ep - gr$. Yet, we still need to be careful that next to information that is about Q not too much other information is also added that is not about Q.

Finally, Cut describes whether smaller elements related to larger relevant elements are also about the same query. Moreover, we know that the smaller element D is about the cut-off information D'. Cut identifies, e.g., the case where a model returns a parent element $D \otimes D'$ about Q and its two children D and D', which share relevant information and are therefore about each each other and the query. Yet, Cut can also mean that $D \Box\!\rightsquigarrow D'$, as they share irrelevant information and $D \otimes D' \Box\!\rightsquigarrow Q$ only because either child is about Q. In order to perform well, models should therefore not unconditionally support Cut but only if it is ensured that the remaining smaller elements are still about the query.

If conditional support for Cut leads to the best performance, models that do not support Cut still perform better than those that support it. Not to support Cut, has the advantage that potentially irrelevant smaller elements that are related to larger relevant elements are never considered to be relevant and can therefore also never be found early in the ranking. This excludes in particular those smaller elements that are less relevant than their related larger element. For $ep-gr$, we do not want to find these early in the ranking. Because $ep-gr$ mainly punishes models that too early return irrelevant elements, it is better not to support Cut than to support it. This is contrary to LMU where we would expect better behaviour from models that support LMU rather than not support it at all. Here, it is more likely that we find the smaller (potentially more focussed) elements earlier in the ranking, while the larger related ones, which contain at least the same relevant information, can be found later.

Next, we look at the performance of each model analysed in Chapter 5 with respect to $ep-gr$. We show how to explain experimental test results using our insights from the theoretical evaluation and with reference to LMU, Mix and Cut. Table 8.1 presents the results for INEX 2005 Thorough $ep-gr$ for the generalised quantisation, on which we would like to concentrate here as an example. In Section 6.3, we introduced the quantisations. Strict quantisation functions are used to evaluate XML retrieval methods with respect to their capability of retrieving highly exhaustive and highly specific components. The generalised functions also reward only fairly relevant elements. We could also see how strict quantisations tend to focus on specificity, while general quantisations favour exhaustivity.

We can clearly see how well most of our models do for these particular tasks. Generally speaking, both models based on proven-to-be-good flat retrieval models, XML vector space and LM I language model, perform well and better than in other evaluation tasks in $ep-gr$ for INEX 2005 Thorough. Gardens Point performs overall not as well in this task as it does in other tasks. This is particularly visible in the generalised quantisations, where it is outperformed by both XML vector space and LM I language model.

For all models in $ep-gr$, we look at the aboutness reasoning properties of LMU, Cut and Mix and what they explain in terms of the experimental outcomes. We start with the best performing model, the XML vector space model.

8.3.2 XML Vector Space

The XML vector space model particularly dominates the system-oriented metrics and generalised quantisation for INEX 2005. It has three approaches submitted for each CO subtask differing in whether they do or do not consider phrases. Regarding the XML vector space retrieval aboutness decision in Section 5.2.4, we developed that $rsv(A,B) = \frac{f(AB)}{||A||*||B||}$, $||.||$ stands for the number of unique tokens, while $f(AB)$ describes a function dependent on the information overlap of A and B. Then, the overall behaviour of $rsv(A, B)$ is determined by the size of $f(AB)$ in relation to the number of unique terms in A and B.

With respect to LMU, we have seen in Section 5.2.4 that the condition on the monotonic behaviour is not very strong, because it does not consider whether the added information is relevant or not but only whether it adds too many unique terms. As the XML vector space model does not discriminate whether new information in a monotonic extension of an element is relevant or not, it is not surprising that, if we find on top of the ranking relevant elements, we also find at least some irrelevant ones. LMU takes care of that. The vector space model does not control monotonic behaviour in that the newly added information has to be relevant but in that it must not reduce the focus, as we have seen in Section 5.2.4.

However, at least XML vector space controls monotonic behaviour in that it prevents the addition of too much unfocussed information. Also, its second AQR step reinforces the results for those elements already relevant by adding terms from those elements as surplus query terms. Considering the effect AQR has on concentrating on the most relevant elements and increasing their ranking, it is therefore not surprising that XML vector space retrieval delivers the most relevant XML elements early in the ranking. This is also the main reason why XML vector space retrieval comes first in comparison with the other models from Chapter 5.

The good $ep - gr$ performance of the model can furthermore be explained by the fact that it does (conditionally) support Mix and does not support Cut according to Section 5.2.4. For XML vector space retrieval, Mix describes, for instance, the case of two children elements merged into their parent. We shall discuss this in more detail in Section 8.4, as some interesting interactions with brute-force filtering follow. As Mix is conditionally supported for XML vector space retrieval, aboutness is preserved for this important characteristics. If children elements are highly relevant, their parents will be, too. Both can then be found on top of the ranking.

As seen in Chapter 5, Mix is supported by all investigated XML retrieval models. Only for vector space retrieval, it is just conditionally given. As just discussed for LMU, this condition on Mix in particular is one of the main reasons why the model outperforms others in $ep - gr$, as it ensured that those elements that contain many relevant but also many less relevant elements are not automatically added early in the ranking. These are, for instance, parent elements of mixed highly relevant and less relevant children. Through the condition on Mix, it is ensured that also this reasoning property does not have undesired side effects if relevant children are displaced from the top of the ranking by their parents.

LMU and Mix are both conditionally given for the XML vector space model. The next model, XML language modelling I, does not offer such a condition and performs overall worse than XML vector space.

8.3.3 XML Language Modelling I

The XML language modelling approach LM I equally performs well for $ep - gr$. For the Thorough task, [Sigurbjörnsson and Kamps, 2005] look at reductions of the number of indexing units with two special indexes: one based on element-length ($UAmsCOTLength-basedIndex$ in Table 8.1) and another one based on past relevance assessments Q_{rel} ($UAmsCOTQrelbasedIndex$ in Table 8.1). These two indexes perform particularly well and come 4th and 5th in the overall assessment ranking.

LMU is fully supported by the LM I approach, which marks one significant difference to the XML vector space retrieval model. This means again that the aboutness decision is preserved across elements that share the same relevant information with smaller elements but also contain other information. Contrary to the XML vector space model, however, no distinction is made regarding the relevance or the focus of the additional information. That LMU is fully supported without control over the newly added information, is one factor why the model performs worse than the vector space one for $ep-gr$.

Looking at LMU, it is also no surprise that Q_{rel} is the model's best submission by some distance. Here, the side effect of unwanted information added by LMU reasoning occurs less likely, as Q_{rel} contains only those elements that according to the experience of previous INEX years are more likely to be relevant.

Mix is fully supported by LM I, too. We have already seen for the vector space model how support for Mix helps explain the overall good performance of XML language modelling for $ep-gr$.

The reasoning behaviour for Cut, on the other hand, helps explain further differences in the submissions of XML language modelling, which are all using different indexes as explained in Section 5.3.3.4. The best performing indexes are the element-length one and the Q_{rel} one, which both exclude elements with little content [Sigurbjörnsson and Kamps, 2005]. Length-based indexing uses only elements with at least 25 words, while Q_{rel} limits the result list to element types from branch elements.

Regarding the results of Thorough task for language modelling, the relative performance of the length-based index and Q_{rel} is strongly improved for the generalized quantisation according to Table 8.1. Both indexes outperform the general language model element-index-based submission ($UAmsCOTElementIndex$), which in most other tasks is the best submission [Sigurbjörnsson and Kamps, 2005]. Our theoretical evaluation offers an explanation. As seen in Section 5.3.3.4, the model supports Cut, as long as not all relevant information is cut away. But, this also means that there are cases when too much relevant information is cut out. In these cases, though the smaller element is still about the query, it is not considered to be still relevant. These elements will, however, appear less frequent, if the set of returned elements is limited to those of a certain size. Thus, the $ep-gr$ ranking is improved for the index.

The XML retrieval models based on standard flat document retrieval models perform well in Thorough. Next, we analyse the submissions of the two models from Section 5.4 specifically designed for XML retrieval.

8.3.4 Gardens Point

The overall performance of Gardens Point is relatively worse for the Thorough tasks than for the Focussed ones [Kazai and Lalmas, 2005], as we shall see in Section 8.4. Though its overall performance is still very good, it is outperformed by several other retrieval models, especially in the generalised quantisations, although its overall performance is still very good. Let us examine again the three reasoning rules we have identified to be decisive for the performance in $ep - gr$. Gardens Point fully supports LMU and Mix with similar consequences as just analysed for language modelling. It also does not support Cut.

According to our theoretical evaluation in Section 5.4.1, the model differs from language modelling and other models in its decay factor $D(c)$ among other things. We can explain the model's behaviour for Thorough with the impact of $D(c)$ on LMU and Mix. $D(c)$ aims to control the impact of parent elements by putting more relevance onto their retrieved children. However, these parent elements might still have a higher retrieval status value than their relevant children and should therefore appear on top according to the ideal ranking. If they do not appear on top in the real submission, they reduce the overall performance in $ep - gr$ significantly.

Using Mix, we can demonstrate the behaviour induced by the factor $D(c)$, which leads to a worse performance. Let us assume that we have a component $D1$ with a retrieval status value of 3 for a query Q and a component $D2$ with a score of 2. Then according to Mix, with $D1 \Box\!\!\rightsquigarrow Q$ and $D2 \Box\!\!\rightsquigarrow Q$ also their parent $D1 \otimes D2 \Box\!\!\rightsquigarrow Q$. Without $D(c)$ the retrieval status value of $D1 \otimes D2$ would be 5, with $D(c)$ it is 2.45, which reduces its rank in the actual submission, increasing its distance from the ideal rank and therefore making worse $ep[r] = \frac{i_{ideal}}{i_{submission}}$.

A similar argument can be made using LMU twice, considering a highly retrieved grandchild and child of a parent. As Gardens Point reduces the retrieval status values of parents with highly retrieved children, it is not surprising that its performance decreases for the Thorough task — particularly in the $ep - gr$ measure. The authors identify this behaviour in their experimental results, but do not offer an explanation:

> 'In the Thorough submission on the other hand, by increasing the value of $D(c)$ we were able to extract more scoring elements from the ancestors of highly scoring leaf elements.'

With our theoretical evaluation, we can derive the impact of $D(c)$ on the $ep - gr$ measure.

8.3.5 Contextualisation Method

For the Contextualisation method, all submissions to Thorough are not among the best according to Table 8.1, but the root one performs best. As seen in Section 5.4.2.1, root contextualisation means that the contextualized weight of an element is a combination of an element's and its root's weights. In the model's submissions the root is weighted by the value 1.5 and then averaged with the actual weight of the element. For the Contextualisation method, we do not want to compare it with the performance of other models, but show how our theoretical evaluation can help explain differences in the several submissions of that one model alone.

According to Section 5.4.2.5, LMU and Mix are both conditionally given for the Contextualisation method with a highly independent condition, while Cut is not at all supported. Contextualisation is special, in that LMU is only conditionally supported by the model, as is has a means of expressing undesired information using negative weights in queries. We have described this in Section 5.4.2.5, and we have also discussed how a threshold might improve the model's performance.

For the $ep - gr$ measure, we have the same query for all elements and want to explain why certain elements are delivered higher in the ranking given a test query Q. These test queries have no negative weights. Thus, Left Monotonic Union is unconditionally given, as we have the same weights for all queries. With LMU, we can conclude that $D1 \otimes D2 \Box\!\!\rightsquigarrow Q$ given that $D1 \Box\!\!\rightsquigarrow Q$. From the fact that LMU is unconditionally given for $ep - gr$ and the fact that Mix is then also unconditionally given, we can derive the conclusion that larger elements, which are about the same query as smaller ones, generally appear higher in the ranking (as it is the case for LM I). For XML documents, those elements closest to the root and the root itself tend to be the largest elements in a document tree. As the largest elements are higher in the ranking and can be found closer to the root, the root contextualisation performs well and better than the tower ones, which combine all ancestor elements.

Cut is generally not supported by the model. This tells us something about the behaviour of smaller elements that are also retrieved compared to larger ones. All models essentially based on information overlap disallow Cut — XML vector space, XML language modelling, but also Contextualisation. It tends to disadvantage smaller elements, which tend to have less information overlap as they contain less information, if the retrieval status values are not normalised as in the XML vector space model. Equation (5.4.3)'s kf_ξ does not seem to include a normalisation, therefore smaller retrieved elements that are related to larger retrieved elements are not returned on top of the ranking. The overall performance under $ep - gr$ becomes worse.

Next, we shall analyse the Focussed task and in particular the complex interaction of filter reasoning with underlying reasoning, as analysed in Chapter 7.

8.4 Focussed

In this section, we consider results from the Focussed task and concentrate on the evaluation using the eXtended Cumulated Gain (XCG) Metrics as the official metrics. These are based on the cumulated gain (CG) based metrics of [Järvelin and Kekäläinen, 2002], which consider the dependency of XML elements (e.g. overlap and near-misses) within the evaluation. The user-oriented measures of normalised extended cumulated gain ($nxCG$) complement the system-oriented effort-precision/gain-recall measures (ep/gr) in INEX 2005, which we have discussed in the Thorough parts in Section 8.3.

Given a rank i, the value of $nxCG(i)$ reflects the relative gain the user accumulated up to that rank, compared to the gain she could have attained if the model would have produced the optimum best ranking. Several other parameters define how overlap is to be handled. Similarly to $ep - gr$, for which models are penalised if they deliver anything else but the most relevant document components first, better performance under $nxCG$ also

Table 8.2: INEX 2005 Focussed with the *nxCG* metric, strict quantisation and rank 10

S.No	Model	RunId	nxCG[10]
4	Gardens Point	3-focussed-highest-VVCAS	0.1324
7	Gardens Point	1-focussed-Leaves-VVCAS	0.1074
8	Gardens Point	2-focussed-highest-VVCAS	0.0959
10	Language model I	UAmsCOFocElements	0.0901
14	Contextualisation Model	Tampere-b09-tower	0.0593
18	XML vector space	focussed-with-phrase...	0.0478
21	XML vector space	focussed-no-phrase...	0.0423
24	Contextualisation Model	Tampere-b09-root	0.0420
28	XML vector space	focussed-no-phrase...	0.0324
29	Contextualisation Model	Tampere-exp5-b09-root	0.0324
31	Language model I	UAmsCOFocSections	0.0231
39	Language model I	UAmsCOFocArticle	0.0077

Table 8.3: INEX 2005 Focussed with the *nxCG* metric, strict quantisation and rank 50

S.No	Model	RunId	nxCG[50]
1	Gardens Point	3-focussed-highest-VVCAS	0.1902
6	XML vector space	focussed-no-phrase...	0.1317
7	Gardens Point	2-focussed-highest-VVCAS	0.1261
8	XML vector space	focussed-no-phrase...	0.1240
11	Contextualisation Model	Tampere-b09-root	0.1137
12	Contextualisation Model	Tampere-exp5-b09-root	0.1094
13	Gardens Point	1-focussed-Leaves-VVCAS	0.1087
17	Language model I	UAmsCOFocElements	0.1014
18	XML vector space	focussed-with-phrase...	0.0918
30	Contextualisation Model	Tampere-b09-tower	0.0442
31	Language model I	UAmsCOFocSections	0.0433
38	Language model I	UAmsCOFocArticle	0.0115

means that models are able to deliver relevant elements first. This means we can rely on similar reasoning rules to Section 8.3.1 to identify the conditions for better performance.

However, under *nxCG* the overall ranking is not as strongly influenced by highly ranked but less relevant document components. Here, it is more important to find all relevant document components in order to steadily increase the gain at each rank. The overall ranking is more important. In the example from Section 8.3 taken from [Kazai and Lalmas, 2005], the second model performs better than the first one under *nxCG*.

This section also brings together the analysis of filters from Chapter 7 with the analysis of Chapter 5's underlying aboutness systems, that are filtered. Filters are the main strategy used in Focussed to provide focussed and non-overlapping results sets. We can now take up our idea from Section 7.2 and investigate how the combination of two aboutness systems changes performance. Starting with the XML vector space model from Section 5.2, we investigate for each XML retrieval model how the filter aboutness behaviour changes the performance of the overall model. We proceed in a similar way we did for Thorough. As in Section 8.3, we look at several key reasoning behaviour properties per model and investigate their impact on the experimental performance.

Table 8.4: INEX 2005 Focussed with the *nxCG* metric, generalised quantisation and rank 50

S.No	Model	RunId	nxCG[50]
1	XML vector space	focussed-no-phrase-...	0.2190
2	Gardens Point	3-focussed-highest-VVCAS	0.2122
8	XML vector space	focussed-no-phrase-...	0.1985
11	Gardens Point	1-focussed-Leaves-VVCAS	0.1870
12	XML vector space	focussed-with-phrase-...	0.1828
15	Gardens Point	2-focussed-highest-VVCAS	0.1756
16	Contextualisation Model	Tampere-exp5-b09-root	0.1691
19	Contextualisation Model	Tampere-b09-root	0.1615
20	Language model I	UAmsCOFocElements	0.1592
23	Language model I	UAmsCOFocSections	0.1531
27	Contextualisation Model	Tampere-b09-tower	0.1243
34	Language model I	UAmsCOFocArticle	0.0988

Table 8.5: INEX 2005 Focussed with the *nxCG* metric, generalised quantisation and rank 10

S.No	Model	RunId	nxCG[10]
2	Gardens Point	3-focussed-highest-VVCAS	0.2561
8	XML vector space	focussed-no-phrase-...	0.2290
9	Gardens Point	1-focussed-Leaves-VVCAS	0.2275
12	XML vector space	focussed-with-phrase-...	0.2214
13	Gardens Point	2-focussed-highest-VVCAS	0.2214
14	XML vector space	focussed-no-phrase-...	0.2163
17	Language model I	UAmsCOFocElements	0.1943
20	Language model I	UAmsCOFocSections	0.1711
21	Contextualisation Model	Tampere-exp5-b09-root	0.1657
22	Language model I	UAmsCOFocArticle	0.1650
23	Contextualisation Model	Tampere-b09-root	0.1648
24	Contextualisation Model	Tampere-b09-tower	0.1648

8.4.1 XML Vector Space

The XML vector space retrieval model has been very successful in the experimental evaluation in INEX 2005 [Mass and Mandelbrod, 2005], but its performance decreases when it comes to returning the single most relevant document component along any path in the INEX 2005 Focussed retrieval task for CO queries.

In order to comply with these requirements, the authors have amended the original model by two additional filters to remove overlapping elements. First a regular Thorough run is performed and then elements are removed in a 'smart filtering' step, in which clusters of highly ranked results in the XML tree are identified. Only the most relevant element in a cluster is picked. A second 'brute-force' step removes all remaining overlap.

One run [Mass and Mandelbrod, 2005] with both stages and a second submission with only the second stage were put forward by the XML vector space model. The submission with both stages performed better. In this section, our aim is to find out why brute-force filtering led to a significant decrease in performance.

Section 7.3.1 has analysed the brute-force filter and shown how it changes the aboutness behaviour of its underlying models by fundamentally changing some of the most important reasoning properties such as Left Monotonic Union. In the case of the XML vector space model, we could show in Section 5.2.4 how, as XML vector space's thresholds are changed, its abilities to adjust to the tasks of Focussed are heavily influenced. In the case of the XML vector space model, potentially advantageous XML reasoning abilities are lost as thresholds are taken out of their context if instead of the N top documents for which $rsv(D, Q) > n$ only those with top scores on a path are taken into account.

XML vector space retrieval successfully uses conditions on Left Monotonic Union reasoning to adjust the behaviour of flat document vector space retrieval to the requirements of XML retrieval under $nxCG$. Especially, the left monotonic behaviour is controlled by disallowing the addition of verbose content, as seen in Section 5.2.4. Yet, this control over left monotonic reasoning is lost for XML vector space modelling, once the brute-force filter is applied, as seen in Section 7.3.1. LMU reasoning is completely eliminated. The XML vector space retrieval model's performance therefore decreases, and it loses its advantage.

The authors have experienced the fundamental change in the reasoning behaviour through the introduction of brute-force filtering in their experimental results. However, they relate it to the overall impact of structural hints in XML retrieval generally instead of the the way their model integrates structural hints. They claim [Mass and Mandelbrod, 2005]:

> 'Structural hints are valuable only when used as a real filter, and not when used merely as recommendations as defined by the CO+S tasks.'

Our theoretical evaluation has delivered another explanation by showing that it might be the particular type of filter that reduces performance, as it changes the reasoning of (among other rules) LMU and its related rules such as Mix.

With LMU, we have just discussed an example of an individual reasoning behaviour that leads to an overall decrease in performance in combination with the brute-force filter. Next, we consider different submissions of the XML vector space retrieval model and support for various user reasoning models from Section 6.4. The model performs better for the generalised quantisations than for the strict ones in Focussed for lower ranks, according to Tables 8.5 and 8.2. As shown in Section 6.5.2, the overall retrieval status value is largely determined by the exhaustivity value for the generalised quantisations, as the exhaustivity value dominates the overall quantisation value.

The improvement for XML vector space retrieval in this exhaustivity-oriented quantisation is significant with an average ranking of 11 in the generalized quantisations for lower ranks compared to 22 in the equivalent ranking for strict quantisation (Tables 8.2 and 8.3). Looking at the results from our theoretical evaluation, we anticipate this kind of better performance for exhaustivity-oriented user quantisations from a model, which is on the one hand side strongly based on a flat document document retrieval model (as seen in Section 5.2) and on the other hand does not discriminate $D \Box\!\!\rightsquigarrow Q$ from $Q \Box\!\!\rightsquigarrow D$, as Symmetry is part of its aboutness reasoning. Still we would not expect such a strong difference, which must have further reasons. Looking back at our analysis of the XML vector space aboutness decision in Section 5.2.4, we think the reason for the dominance of the exhaustivity-oriented quantisation lies in that fact that the document components weights clearly dominate in the main rsv function 5.2.1:

$$rsv(D,Q) = \frac{\sum_{t_i \in \{Q \cap D\}} w_Q(t_i) * w_d(t_i) * idf(t_i)}{\|Q\| * \|D\|}$$

In $rsv(D,Q)$, D dominates for both for the numerator and the denominator. Particularly, the number of unique terms in the document component $\|D\|$ is generally much larger than the number of unique terms in the query $\|Q\|$. The composition of the document component has thus a much stronger influence on whether the threshold is passed. The model therefore performs better for those user agents that are concentrated on the exhaustivity dimension, as the measure is here on how much information is covered in D_{ex} according to Section 3.3.3.

If we consider filters to be another aboutness decision in themselves, we can see how they fundamentally change the original aboutness decision with new rules and reflection properties. Already [Wong et al., 2001] have identified as one of the biggest advantages of a theoretical evaluation that it is more open to debate, as underlying assumptions (of an IR model's performance) can be sometimes hidden by the overall mathematical models. Such new transparency leads to new insights about the behaviour of models in general and not only for particular evaluation tasks. For the XML vector space model, this means that we can disclose its underlying reasoning assumptions by identifying the filter as an additional reasoning step, which is external to the original XML vector space aboutness reasoning. We can therefore open up the debate on the general usefulness and configuration of such filters.

8.4.2 XML Language Modelling I

For language modelling I, we have identified two advantages compared to the vector space model in Section 5.3. Firstly, some structural context of an XML element (though limited) is taken into account in the aboutness decision, because the language model is combined with collection and document model. Secondly, the threshold in the aboutness decision is internal. Nonetheless, as this threshold is only related to the overall collection language model, the aboutness decision is still derived from the overlap of information in document components and query, and not really considering structure. We concluded in Section 5.3 that structure is only indirectly considered in XML language modelling.

Language modelling I [Sigurbjörnsson and Kamps, 2005], had three runs in the INEX 2005 Focussed task:

1. $UAmsCOFocArticle$ is a baseline submission created using the article index.

2. $UAmsCOFocSections$ uses a mixture model of the section index and the article index.

3. $UAmsCOFocElements$ is a submission created using a mixture model of the overlapping element index and the article index. Overlap is removed by going through the index list and removing elements overlapping with an element appearing previ-

ously in the list. This is a type of brute-force filter. It is this run we would like to concentrate on.

According to Tables 8.2 and 8.5, LM I performs well for the elements' index in lower ranks but not for section and article indexes. In higher ranks (Tables 8.3 and 8.4) even the performance for the elements' index falls behind the performance of other models [Kazai and Lalmas, 2005]. For LM I, we would like to answer in more detail why its performance is worst for higher ranks and relate this to the way structure is not considered in its aboutness decision. Furthermore, we would like to concentrate on the performance decrease in the element-based index, as the overall best submission for LM I.

For the lower ranks of the INEX 2005 Focused task, it is not surprising that the best performing run for the language modelling I approach is the element-based one. As according to Section 5.3.2.2 LMU is fully supported, it returns all the most relevant elements first — independent of whether they are larger articles or sections.

The elimination of LMU reasoning through brute-force filtering has to lead to a decrease in performance for higher ranks (Tables 8.4 and 8.3). Here, we would find those XML elements that overlap in information with more relevant and therefore higher ranked elements. That such also relevant elements are eliminated must lead to a performance decrease under $nxCG$, as we have also observed for XML vector space retrieval. Under $nxCG$, larger elements that contain relevant and irrelevant information at the same time are expected to be found later in the ranking, where they, however, would still add to the expected gain of information later in the ranking and therefore improve performance.

Next to such general reasons for a performance decrease, it is also important to add that LM I does not use the XML structure in its aboutness decision. LM I performs worse for higher ranks, as it is not able to deliver in the lower ranks those elements that have similar content to highly ranked XML elements but might be on a different XML path. As seen in Section 5.3.2.2, LM I uses structure only to allocate elements into several different indexes and does not use it in the actual aboutness decision. The brute-force filter for $UAmsCOFocElements$, which is applied on top of the language modelling aboutness model and thus intersects with it according to Section 7.3.1.3, is based solely on the overlap in information and not the structural relatedness of two elements. This has more negative consequences for the performance under $nxCG$, as we shall explain next.

In order to elucidate the negative impact of brute-force filtering intersecting with a model essentially based on unconditional information overlap aboutness such as LM I, let us assume that we have a document component $D1$ and a document component $D2$ on two XML paths. Let $D1$ have achieved a higher score for LM I than $D2$. Let us further assume that $D1 \equiv D2 \otimes D2'$. Then, both $D1$ and $D2$ is about the same query Q and can be found somewhere in the element index. That $D1$ and $D2$ are structurally different only plays a role if they would also be allocated into different indexes. The element index on the other hand combines all elements. Using brute-force filtering, once $D1$ is traversed in the lower ranks, $D2$ is removed from the index, too. This means that it is not delivered anymore as an alternative answer and $nxCG$ performance decreases for higher ranks. This effect is noticed particularly in higher ranks, as the relevant XML elements, which have been eliminated through the brute-force filter, would have been found here.

Further experimental performance issues for a model based on information overlap can also be seen by comparing the generalised quantisations in Tables 8.5 and 8.4 with the strict quantisations in Tables 8.2 and 8.3. For all these, the performance of LM I is similarly not among the best, but the performance is on average better for the exhaustivity-oriented quantisations. We have just discussed for XML vector space retrieval that we expect a better support for exhaustivity-oriented users from a model that includes Symmetry, Transitivity, etc. in its aboutness reasoning. In fact, also for LM I, which has a similar reasoning behaviour for these rules to the XML vector space model, exhaustivity-oriented users seem to be better served. As document components, however, do not dominate its retrieval status $P(t_i|e)$ in Section 5.3.3.1 as much, the difference is not as significant as for XML vector space retrieval.

The next model Gardens Point is closer to pure type reasoning according to Section 5.4.1. We will see how this expresses itself in its performance.

8.4.3 Gardens Point

The Gardens Point model is overall a top performer at INEX 2005. It performs better for the specificity-oriented strict quantisations, as it attempts to imitate the mathematical relation for the specificity assessment. We could show in Section 5.4.1 how close its aboutness decision is to the aim of retrieving just those document components, which contain only relevant information. It judges each document component on the basis whether it contains distinctive query terms and only considers those to be relevant for the overall aboutness decision. According to Section 5.4.1, its aboutness decision is based on those parts of XML elements that are also part of the query, which is close to the character counting method that determines specificity in INEX 2005 (Section 6.5.2).

Gardens Point makes use of the fact that in INEX 2005 for the first time, specificity is described in purely mathematical terms using a continuous scale and a counting measure for the overlap in XML element and query. It expresses this mathematical relationship in a simple and clear calculation for aboutness, which is based on a sum of those terms overlapping in document component and query. Yet, Gardens Point performs almost equally well in Focussed for the exhaustivity-oriented generalized quantisations, as it has for the specificity-oriented strict quantisations (Tables 8.2 and 8.3). This good support for exhaustivity-oriented users must have further reasons than for XML vector space and LM I, as Gardens Point does not include Symmetry reasoning in its aboutness decision.

We believe Gardens Point performs well under all quantisations in the Focussed task, as the impact of query and document is the same in its simple aboutness decision for leaf elements in Equation (5.4.1): $rsv_L = K^{n-1} \sum_{i=1}^{n} \frac{t_i}{f_i}$. This contrasts to XML vector space retrieval, where exhaustivity-oriented users and document components are advantaged. Please recall our analysis from Section 6.4, where we defined D_{ex} to be the subsituation that determines a component to be an exhaustive answer, and Q_{sp} to be the one that makes a specific answer. Gardens Point is obviously equally able to identify both Q_{sp} and D_{ex}, as its aboutness decision (according to Equation (5.4.1)) is mainly influenced by t_i, which is defined as those terms common to Q and D.

This explains the equally good performance of Gardens Point for all quantisations. Next, we investigate why the model performs slightly worse for higher ranks in Tables 8.4 and 8.3, which is an indicator for the impact of $D(c)$ on branch elements according to Equation (5.4.2), and again shows how a detailed theoretical analysis can open up the debate on some reasoning assumptions in models, which might be otherwise hidden in the overall mathematical calculations.

As analysed in Section 5.4.1.5, with its parameter $D(c)$, Gardens Point does not penalise enough branch elements with many retrieved children. These become more important for higher ranks, as in those higher ranks exactly those branch elements with more retrieved children will appear. They have been eliminated from the lower ranks by the emphasis on their retrieved children through $D(c)$. However, they still appear in the higher ranks, where the impact of $D(c)$ is outperformed by their higher retrieval status values.

[Geva, 2005] takes notice of the potential issues with $D(c)$ in Focussed:

> 'In the focussed retrieval task it became clear from both qualitative analysis and from experimentation with 2004 data, that it would be advantageous to select elements slightly higher in the tree than the leaves – on account of increased exhaustivity – but not too high since specificity tends to drop. We could control the bias towards the leaves or the internal nodes by increasing or decreasing the decay factor for score propagation. By choosing smaller values for $D(c)$ we were able to increase the relative scores of leaf elements.'

With our theoretical evaluation, we are able to provide a possible explanation for the impact of $D(c)$ on the aboutness behaviour. We expected this behaviour for higher ranks according to our analysis of the monotonic behaviour of Gardens Point in Section 5.4.1.5.

Gardens Point fully supports Mix reasoning. We can generally say that Mix reasoning elimination through brute-force filtering adds to a performance decrease for the focussed tasks. In order to exemplify desired reasoning behaviour that is eliminated with Mix, let us consider the following example: Among other things, Mix describes that, if two children $D1$ and $D2$ are about a query, then their parent item $D1 \otimes D2$ is also about the same query. This behaviour is typical to XML-based reasoning. If this behaviour is not supported, problems might arise, such as the elimination of potentially highly relevant children. Say, we have one relevant child and a more relevant parent, then the child is eliminated from the result set after applying brute-force filtering. Another child of the same parent that is about the same query, is also eliminated, as the parent is already chosen for its path. However, this child might be highly relevant, too.

8.4.4 Contextualisation Method

In the Contextualisation model's INEX 2005 submissions, the contextualisation step improves performance [Kazai and Lalmas, 2005], in particular root and tower contextualisation. As discussed in Section 5.4.2.1, tower contextualisation is an average of the weights of an element's ancestors. Root contextualisation means that the contextualised weight of an element is a combination of the weight of an element and its root. The parent contextualisation, for which only the direct parent is considered, delivers the smallest improvement, as it offers too small a context [Arvola et al., 2005b].

The model's Focussed experimental performance delivers mixed results. The tower contextualisation performs well for lower ranks, better than XML vector space retrieval. However, we can clearly see in Tables 8.4 and 8.3 that the performance of the tower contextualisation decreases for higher ranks and falls behind XML vector space retrieval. The performance of the root contextualisation remains roughly the same. Its ranking is improved, as the performance of other models is falling further behind. These mixed results are linked to the way overlap is removed in the model and how those elements that have similar content but are on different paths in the XML document tree are removed from the results.

Let us consider an example and assume that an element $D1$ and its nephew $D2$ are both about Q. Let us further assume on its path to the root $D1$ has the highest relevance, while for $D2$ its ancestor $D3$ has a higher relevance. $D3$ is also the parent of $D1$, but less relevant than $D1$. In the attempt to remove overlap $D3$ will have been removed from the result list, as it is less relevant than $D1$. For its path $D2$ is therefore returned, as the higher ranking $D3$ has been removed. This leads to the worse performance for higher ranks in the tower contextualisation, where the contextualisation step further punishes those elements that have lost more relevant ancestors such as $D1$ through brute-force filtering. This explains tower contextualisation's decrease in performance for higher ranks. The root contextualisation is not affected by this, as the root is the same for all elements in a document tree.

We have examined the reasoning behaviour that leads to this kind of worse performance, while looking at the Cut aboutness reasoning for the Contextualisation model in Section 5.4.2.5. Cut assumes that $S \Box\!\!\rightsquigarrow T$, with $S \otimes T \Box\!\!\rightsquigarrow U$ and $S \Box\!\!\rightsquigarrow T$. Let S be $D2$ from the example above, U be Q and $D1$ be T. Then, according to our assumption that $D1$ and $D2$ contain similar information: $D1 \Box\!\!\rightsquigarrow D2$. Therefore the two assumptions of Cut are given: $D1 \otimes D2 \Box\!\!\rightsquigarrow Q$ and $D1 \Box\!\!\rightsquigarrow D2$. As Cut is not part of the aboutness reasoning for the Contextualisation model, we cannot conclude that $D1 \Box\!\!\rightsquigarrow Q$. This describes the experience from the experimental behaviour, we have just analysed. It also shows that elements that are subsituations of other elements, which are about a particular query, do not necessarily have to be about the same query. This becomes more important when we look at the behaviour of the Contextualisation model for the generalised quantisations next.

According to Tables 8.5 and 8.2, the Contextualisation model performs worse for the generalized quantisations in lower ranks than for the strict quantisations. Let us return to Equation (5.4.3): $w(k, \xi) = \frac{kf_\xi}{kf_\xi + v \times ((1-b) + b\frac{\zeta f_c}{\zeta f_k})} \times \frac{log(\frac{N}{m})}{log(N)}$. In the model large values for b are used in the weighting scheme to eliminate larger elements for focussed tasks [Arvola et al., 2005b]. Larger elements are generally closer to the root in an XML document tree. Instead of these elements, subsituations of them or elements closer to the leaves are chosen. We have, however, just seen that these do not necessarily have to be about the same query, as

Table 8.6: INEX 2005 FetchBrowse with the $ep - gr$ metric and generalised quantisation

S.No	Affiliation	RunId	MAep
3	Gardens Point	1-FetchBrowse-VVCAS	0.0850
4	Gardens Point	2-FetchBrowse-SVCAS	0.0850
9	XML vector space	no-phrase-...	0.0573
12	XML vector space	no-phrase-...	0.0453
15	XML vector space	no-phrase-...	0.0399
23	Contextualisation model	Tampere-FB-b09	0.0172
25	Contextualisation model	Tampere-FB-b09-parent	0.0147
26	Contextualisation model	Tampere-FB-b09-tower	0.0140
29	Gardens Point	QUT 3-FetchBrowse-focussed	0.0093
32	Language model I	UAmsCOFBElements	0.0056
34	Language model I	UAmsCOFBSections	0.0033
38	Language model I	UAmsCOFBArticle	0.0016

Cut is not supported, which explains the loss in the ability to return the most exhaustive elements in the generalised quantisations.

In the final section of this chapter, we now analyse FetchBrowse as an indication of the ability to correctly identify elements within relevant documents.

8.5 Fetch & Browse (FetchBrowse)

The FetchBrowse task [Gövert et al., 2006] is inspired by the work of Chiaramella in [Chiaramella, 2001]. In the task, at first relevant articles are identified in a fetching step in order to afterwards find in the browsing step the most exhaustive and specific elements within those fetched articles. Both steps produce rankings according to the two evaluation dimensions of exhaustivity and specificity, once for the articles in a collection and then for the elements within these articles. In the end, we have an article-level and an element-level retrieval status value.

Though inspired by it, FetchBrowse is quite different from [Chiaramella, 2001], which we used to define aboutness for exhaustivity and specificity. Here, fetching in particular is not limited to article elements but is done over the complete element base. In the browsing step, a compromise is then sought between most effective exhaustivity and specificity. We have described this in Section 3.3.1. In INEX 2005, FetchBrowse retrieval is oriented towards the user. Ranked documents are the output, together with all relevant elements within those documents. This simulates a user browsing for the most relevant elements within relevant articles.

For FetchBrowse, in the fetching phase all articles D_{art} about a query Q are returned: $D_{art} \Box\!\!\rightsquigarrow Q$. Then, in the browsing step any relevant element D_k is returned: $D_k \Box\!\!\rightsquigarrow Q$ and $Q \Box\!\!\rightsquigarrow D_k$, where $D_{art} \equiv D1 \otimes ... \otimes D_k \otimes ... \otimes D_n$. For the FetchBrowse task, not only the queries are the same but also the documents components, as we consider each article D_{art} separately. This means that, for this retrieval task, the document components are all related and are all about the query in scope. They differ, however, in size and in their relevance. So, the task investigated by FetchBrowse is the ability to split up an article situation into its relevant subsituations, while at the same time avoiding those subsituations that are not relevant. As the elements only differ in size but not in their

Table 8.7: INEX 2005 FetchBrowse with the $ep - gr$ metric and generalised quantization

S.No	Affiliation	RunId	MAep
7	XML vector space	no-phrase-...	0.0180
8	XML vector space	no-phrase-...	0.0176
9	Gardens Point	1-FetchBrowse-VVCAS	0.0164
10	Gardens Point	2-FetchBrowse-SVCAS	0.0164
15	XML vector space	no-phrase-...	0.0121
22	Contextualisation model	Tampere-FB-b09-tower	0.0072
23	Contextualisation model	Tampere-FB-b09-parent	0.0071
24	Contextualisation model	Tampere-FB-b09	0.0071
27	Gardens Point	3-FetchBrowse-focussed	0.0044
29	Language model I	UAmsCOFBElements	0.0023
34	Language model I	UAmsCOFBSections	0.0013
40	Language model I	UAmsCOFBArticle	0.0002

aboutness relation, we return again to the monotonic behaviour and how it helps to identify the right subsituations.

8.5.1 XML Vector Space

Let us once more commence with the XML vector space retrieval model. It uses a straightforward approach to implement the FetchBrowse task [Mass and Mandelbrod, 2005]. First a standard Thorough submission is run, without any filter. In a second step, the relevant articles are identified. Within these, elements are ranked according to their retrieval status value.

As according to Tables 8.6 and 8.7 XML vector space retrieval performs overall better for the generalized quantisations, the interesting question seems to be why it is able to provide a better return for exhaustivity-oriented users than for specificity-oriented users in FetchBrowse. This behaviour is linked to the model's ability to influence aboutness of larger elements (or larger subsituations) by its conditional support for LMU, which supports a good performance under the generalised quantisations, as already discussed in Section 8.3.2. At the same time Cut reasoning is not supported by the model (according to Section 5.2.4), which means smaller relevant subsituations are not necessarily about a query if their larger relatives are.

XML vector space retrieval therefore supports to retrieve those relevant subsituations that are exhaustively about a query, as they are larger, while it does not support well the return of smaller, potentially more specific subsituations. Its performance has to be better for the generalised quantisations.

The worst performing model for FetchBrowse is with some distance LM I language modelling. We will not further analyse it here, as it has reported some major problems with its submission for FetchBrowse. Instead, we concentrate on the Gardens Point and the Contextualisation models.

8.5.2 Gardens Point

Gardens Point's approach to FetchBrowse includes sorting the set of relevant subsituations by article relevance and afterwards per article by element relevance [Geva, 2005]. Contrary to XML vector space retrieval, Gardens Point performs worst for all its submissions for FetchBrowse generalized but outperforms all other models in scope in the strict quantisation according to Tables 8.6 and 8.7. This is interesting, as Gardens Point fully supports Left Monotonic Union, which means that always with $D \Box\!\!\rightsquigarrow Q$ also $D1 \otimes D2 \Box\!\!\rightsquigarrow Q$. A support for this kind of reasoning would lead us to expect a worse performance for the strict quantisation for Gardens Point, as elements with more children and therefore larger and closer to the root benefit more from full support for LMU.

So, why is Gardens Point's performance then so good for the strict quantisations? We relate this to the fact that right monotonic reasoning is not supported. For the strict quantisation, we are interested in the specificity-oriented user return of $Q \Box\!\!\rightsquigarrow D$. Our interest is $Q \Box\!\!\rightsquigarrow D_k$, where $D_{art} \equiv D1 \otimes ... \otimes D_k \otimes ... \otimes D_n$. According to Section 8.5, this describes the browsing step. As Gardens Point does not support Right Monotonic Union, it disallows the growth of relevant document components to $D_k \otimes D'$, as it is not guaranteed that $Q \Box\!\!\rightsquigarrow D_k \otimes D'$.

XML vector space retrieval at least conditionally supports Right Monotonic Union and can therefore not prevent the inclusion of undesired subsituations in the answer set, which comparably lowers its performance for strict quantisations in FetchBrowse. Gardens Point allows right monotonic behaviour only when it is safe and should be supported in the case of Context-Free And. According to Context-Free And, from $Q \Box\!\!\rightsquigarrow D1$ we can only say that $Q \Box\!\!\rightsquigarrow D1 \otimes D2$ if also $Q \Box\!\!\rightsquigarrow D2$. This means the added element information must be relevant.

Gardens Point seems to particularly perform in the strict quantisations also in the FetchBrowse task, as it is the only analysed model that does not support Right Monotonic Union and is therefore at least for this reasoning ability closer to pure type XML retrieval. In Section 4.7.4 we showed that pure type XML retrieval does not support Right Monotonic Union, as with a change in the composition of the query, the hierarchical inclusion is changed, too.

8.5.3 Contextualisation Method

For the FetchBrowse task, Contextualisation also fully supports monotonic reasoning, as just like for Focussed its ability to control monotonic behaviour by using negative weights to identify undesired elements does not apply. For Contextualisation, we are interested in finding out why the contextualisation step itself does not seem to have an impact on FetchBrowse performance. According to Tables 8.6 and 8.7, the Contextualisation runs are in both quantisations very close to each other, which we take as an indication that the actual contextualisation step is not very useful to discriminate relevant subsituations of different size within the same article, which is tested in FetchBrowse.

We explain this with the fact that the contextualisation steps are based on averaging an element's weight with either some or all of its ancestors [Arvola et al., 2005b]. The method of taking an average does not seem sufficient to influence the ranking of elements, if differences in estimated relevance are mainly linked to differences in element sizes for FetchBrowse. This is the case as the average is taken over related elements, over children and their ancestors, which must be counter-productive for FetchBrowse. It must make it more difficult to determine which of the elements that are part of the average calculations are the most relevant ones.

8.6 Conclusion

In this chapter, we showed how results from the theoretical evaluation help explain experimental results. Such work can obviously not cover all possible explanations but has to focus on certain significant properties coming out of the experimental evaluation.

For the experimental evaluation of the INEX 2005 Thorough task we derived how Left Monotonic Union, Mix and Cut influence a good performance. For the remaining two INEX 2005 tasks, Focussed and FetchBrowse, we determined which reasoning properties support a good performance. For the Focussed task, in particular, we investigated how filter reasoning and underlying reasoning behaviour work together. We could see how especially the changes to the monotonic reasoning behaviour enforced by the brute-force filter influence retrieval performance.

We left out other significant results from our theoretical evaluations in Chapter 5, as we wanted to concentrate on demonstrating the particular importance of monotonic reasoning for experimental performance. Containment is one example for a reasoning property we did not consider in this chapter. It is significant, as only XML vector space retrieval does not support it and therefore mimics the behaviour of pure type XML retrieval in this case (see Table 5.10). We would expect this to contribute to its overall convincing performance.

Containment states that if two subsituations contain each other, their parent situations are also about each other. From $S_i \rightarrow T_i$, we can conclude that $S \Box\!\!\rightsquigarrow T$. This is not necessarily a desirable reasoning characteristic in XML retrieval. Let us assume that $S_i \otimes S_j \equiv S$. Then, S_j might well contain a lot of information that is not about T, which makes S less relevant and less focussed to T. This example shows that (again depending on the experimental task) a full support for Containment is not necessarily desirable.

We could go on with other results from Chapter 5 but as we said at the beginning of this chapter, we can only ever provide a snapshot of possible explanations that are derived from the theoretical evaluation and help understand the experimental evaluation. As there are potentially many results from an experimental evaluation we can never offer all the explanations from a theoretical point of view.

In this chapter monotonic reasoning has proven once more to be key to a model's success. In many ways, the importance of monotonic reasoning is also reflected in the successful work of [Fang et al., 2004], where the authors work with retrieval heuristics and basic desirable constraints that any reasonable retrieval function needs to satisfy for good retrieval performance. We come back to a discussion of monotonic reasoning in the concluding chapter of this thesis.

Chapter 9

Conclusion and Future Work

In this concluding chapter, we list the main contributions this thesis has made and our conclusions. We also present some perceived limitations of our current approach as well as how future work might address these and amend our current framework.

9.1 Contributions

This thesis has delivered a new theoretical framework to analyse XML retrieval based on aboutness approaches to theoretical evaluation. We believe that a theoretical evaluation approach is particularly suitable for the complex XML retrieval tasks, as its challenges are directly linked to difficulties in describing the complex nature of the interaction of structure and content in XML retrieval. Our approach offers a methodology to bring together XML content and structure as well as reasoning behaviour using them into a unified framework.

Our contributions can be broadly summarised as firstly those that developed a theoretical evaluation methodology for XML retrieval. These include the definition of XML aboutness as well as the refinement of existing theoretical evaluation methodologies to match the requirements of XML retrieval. The second main set of contributions stems from the evaluation of actual XML retrieval models in INEX. We could identify important reasoning properties that help with good XML retrieval performance as well as translation and adjustment strategies used to redefine flat document retrieval for use in XML retrieval. Finally, with the analysis of filters, the explanation of experimental results and the evaluation of XML retrieval experimental evaluation methodologies, we have intervened in the discussions in the INEX XML retrieval community.

9.1.1 Theoretical Evaluation Methodology for XML Retrieval

This thesis has presented a framework based on aboutness that allows to analyse the characteristics of particular XML retrieval models. We have shown that existing results of an aboutness-based theoretical evaluation in flat document IR indicate that it can be a powerful methodology to also analyse the more complex tasks in XML retrieval. Our hypothesis has been that particularly in the domain of structured document retrieval, an aboutness-based theoretical evaluation presents a powerful methodology to analyse how the inclusion of XML structure in the aboutness decision leads to the best performances in the experimental evaluation. In order to support this hypothesis, we have first developed a new aboutness criterion to match the requirements of XML retrieval, and have then delivered a new methodology to analyse XML retrieval models based on this new aboutness criterion.

9.1.1.1 XML Retrieval Aboutness Criterion

The main contribution of Chapter 3 is the definition of Situation Theory aboutness for XML retrieval in Sections 3.3.2 and 3.3.3. We have shown how [Chiaramella, 2001] and [Nie, 1988] have used the conditional $d \rightarrow q$ by van Rijsbergen to model the general relevance (exhaustivity) of an element to an information need, and added to this a second one $q \rightarrow d$ to model the focus of an element (specificity). In our framework, this distinction becomes $D \Box\!\!\rightsquigarrow Q$ or D about Q for exhaustivity and $Q \Box\!\!\rightsquigarrow D$ or Q about D for specificity. Only those XML retrieval models able to differentiate the left and the right hand side of $\Box\!\!\rightsquigarrow$ can also make a difference between $D \Box\!\!\rightsquigarrow Q$ for exhaustivity and $Q \Box\!\!\rightsquigarrow D$ for specificity.

Further to the introduction of the second dimension $Q \Box\!\!\rightsquigarrow D$, we also needed to amend existing Situation Theory definitions of aboutness. Standard Situation Theory aboutness claims that situation S is about a situation T if and only if T contains one infon i such that situation S is about infon i. This works well for flat document retrieval models. It, however, leads to problems if we look at XML retrieval aboutness. The one common infon i could be an infon expressing structure, possibly itself bearing no information useful to a user. In XML retrieval, two XML situations could share the same infons expressing structure, as they share the same document type definition. Structure in text-centric XML only supports meaning but does not create meaning. Therefore, we needed to find another, stricter aboutness criterion that uses 'subsituations' instead of simple infons (Section 3.3). A subsituation is a situation S_i that is part of another situation S with content and therefore meaning. In order to conclude that a property like INEX specificity does not apply to a situation S, we just need to show that a situation with that property cannot be a subsituation of S.

9.1.1.2 XML Retrieval Evaluation Methodology

Building on the XML aboutness criterion, we have offered a new theoretical evaluation methodology to analyse XML retrieval. Our theoretical methodology works through four steps. Section 4.1 has introduced the first three steps that our framework shares with those that analyse flat document retrieval models. In the first translation step a formalism is delivered to express aboutness symbolically. The second step specifies aboutness by deriving rules of reasoning behaviour. It uses a set of reasoning *rules* to describe the functional behaviour of the XML retrieval aboutness and to discriminate specificity and general relevance reasoning (Section 4.4). The third step derives a reflection of aboutness boundaries (Section 4.5). It defines typical non-reasoning related boundary elements of retrieval systems.

Section 4.7 has presented the forth step, which is specific to XML retrieval: the pure type XML retrieval model to capture the influence of XML structure on aboutness. While the first three steps have been taken from the work of Huibers and others and adjusted to the requirements of XML retrieval, pure type XML retrieval is our addition in order to qualify the impact of XML structure on aboutness behaviour.

In order to develop pure type XML retrieval, we first needed its aboutness decision. To this end, we have developed hierarchical inclusion in Section 4.7.1 as an expression of the fact that XML enforces a hierarchical representation of a document, as elements are organised into a tree structure. We finally needed a workable definition of pure type XML retrieval aboutness using our Situation Theory framework. To this end, Section 4.7.2.1

has defined a set-based translation.

In Section 4.7, we were able to derive translation and reasoning properties for pure type XML retrieval, which we could then use in Chapter 5 to analyse the impact of XML structure on the reasoning behaviour of individual XML retrieval models.

9.1.2 Evaluation of XML Retrieval Models

In Chapter 5, all the steps of the theoretical evaluation are applied to XML retrieval models, which have been successful in the INEX evaluation. We have concentrated on successful models, in order to demonstrate that we can show differences in models that are mature. We do not want to repeat here individual results for particular models, but rather concentrate on some re-occurring topics we found to be relevant for the analysis of all XML retrieval models. We cover first those reasoning properties that have proven to be important for XML retrieval.

9.1.2.1 Important Reasoning Properties

In our theoretical evaluation of XML retrieval models, we could see how XML retrieval work is concentrated on the control of monotonic behaviour and other reasoning properties like Symmetry, which heavily influence the primary aim of XML retrieval aboutness decisions to find the most focussed answer. For instance, a full support for Left Monotonic Union can be counterproductive for the identification of the right level of granularity. If an XML element D is about a query so will be its parent $D \otimes D1$ according to LMU. However, in XML retrieval we would like to make exactly this distinction between D and $D \otimes D1$.

Almost none of the analysed XML retrieval models is close to the monotonic reasoning exhibited by pure type XML retrieval. No model supports Cut reasoning and can therefore maintain aboutness if larger relevant elements that are about a query are reduced to smaller ones. It is also very interesting that all but the Gardens Point model support right monotonic reasoning. RMU does not necessarily support better retrieval results. RMU allows us to conclude from the assumption $D \Box\!\!\rightsquigarrow Q$ that also $D \Box\!\!\rightsquigarrow Q \otimes Q'$. However, in XML retrieval Q might well include a structural condition. For instance, Q alone might point to a section while $Q \otimes Q'$ might point to a paragraph within a section, which would completely change the aboutness relation.

It is this kind of desirable behaviour that implies that XML retrieval systems should be able to change an aboutness decision if the XML context changes. This entails that the non-monotonic reasoning rules we have presented in Section 4.4 are a good foundation for the theoretical analysis of XML retrieval systems. They allow to describe aboutness as a (non-)monotonic reasoning function with various variables that often include terms and their frequency values, but also other parameters. The description of the monotonic reasoning behavior of XML retrieval models is key to the distinction of flat document retrieval.

9.1.2.2 Relationship to Flat Document Retrieval

Even for the models specifically designed for XML retrieval and discussed in Section 5.4, we found many commonalities with underlying flat document retrieval models. This has been particularly apparent when looking at the translations and reflections of the models analysed in Chapter 5.

For the translation, the main difference to the underlying flat document retrieval model was often that XML elements were indexed instead of full documents. The individual elements, however, were taken to be independent of each other. Even models that consider the XML relationship between elements in Chapter 5 do not do so directly. Structure is not considered in itself but as a relationship between content in XML documents. This can be done, as there is the direct relationship between content components of an XML document and its corresponding XML tree: If in a document D a document component $D1$ is contained by component $D2$ then in the corresponding XML tree $D1$ will be a descendant of $D2$, etc.

This content relationship is used in the language modelling approaches from Section 5.3 if language models of XML elements are interpolated, but also in Gardens Point. Thus, in Section 5.4.1.5 we could see the typical approach in XML retrieval that combines the evidence from XML structure with content relationships known from flat document retrieval. Gardens Point goes furthest in this approach and therefore is very successful in INEX.

For all models in Chapter 5, the reflections differ heavily from the reflection of pure type XML retrieval. All models are not able to discriminate the behaviour for the cases where we find bottom exhaustive and specific document components and queries. No model has developed a concept of top specific document components, which would be a theoretical version of a document component that is always a focussed answer. These important boundaries elements are left out by all models, and we could see how this has an impact on performance.

9.1.2.3 Adjustments

It has become apparent in our theoretical evaluations that most XML retrieval models are based on successful flat document retrieval models and adjust them to the new requirements of XML. Thresholds then seem to have been the most successful way of adjusting the behaviour of retrieval models to the requirements of XML retrieval. Others like the interpolation of the relevance of elements with the one of their ancestors have been less convincing. Throughout this thesis, we have seen how thresholds at various levels of the aboutness decision might improve the performance.

We have identified two types of thresholds, internal ones and external ones. The XML vector space retrieval model has an external threshold, chosen a priori. Here, the threshold has been successfully used to adjust the (monotonic) reasoning behaviour, which has in turn led to a better experimental performance.

Language Modelling I (LM I) is also based on a thresholded aboutness decision. This time, however, the threshold is internal and derived from the collection and article context. In the vector space model the threshold functions as a means to control aboutness behaviour, while here it is used to avoid undesired side effects in language modelling approaches.

9.1.3 INEX Specifics

This section looks to summarise the evaluation results of specific developments within the INEX evaluation campaign. It summarises results from Chapters 6, 7 and 8, which have covered the evaluation of XML retrieval evaluation, the analysis of XML retrieval filters and finally the experimental evaluation in INEX.

9.1.3.1 Evaluation of XML Retrieval Evaluation

Chapter 6 has offered a new perspective using the possibilities of theoretical evaluation. We presented a theoretical evaluation of existing experimental evaluations. Our subsituation-based aboutness criterion led us to an integrated model for user expectations and assessment methodologies in INEX 2004 and 2005. We could first represent how different INEX quantisations express user expectations and use Situation Theory to formalise these expectations as reasoning processes. In a second step, we were able to relate these user models to the INEX evaluation scales and show what patterns of reasoning are involved in these.

Finally, we have pointed at a theoretically consistent alternative treatment of exhaustivity and specificity for INEX 2005 and have suggested to consider both not as independent values, but as based on the same relevant information. We have shown how to strengthen the exhaustivity judgment in INEX 2005 by applying the same mathematical rigour to it as to specificity. We suggested to look at $exh = \frac{|D_{ex}|}{|Q|}$, as probably a better measure for exhaustivity than the INEX 2005 scale. By using the same highlighting for exhaustivity that was used for specificity, we have theoretically demonstrated that it is possible to look at exhaustivity and specificity as two views of the same aboutness property and not as two different aboutness relations.

9.1.3.2 Filters as second-layer Aboutness Decisions

In Chapter 7, we concentrated on filters in INEX and how they attempt to deliver specificity aboutness. Filters are the predominant form in INEX to achieve most focussed answers in retrieval. We looked at how filters for focussed retrieval have an impact on aboutness behaviour of the underlying aboutness system they are filtering. We introduced filters as a second aboutness reasoning on top of an underlying aboutness reasoning specific to the model. Then, the question is whether the two types of aboutness reasoning are in accord with each other.

In order to answer this question, we have developed a new methodology that allows us to formally relate filter aboutness decisions to the ones of the underlying aboutness system. Our theoretical analysis of filters has been done in three steps. We have first formalised the translation process. Secondly, we have identified the reasoning rules associated with the filter. Finally, we have analysed the relationship between the filter and the underlying aboutness systems. For the latter, we have made use of the filtering function f-*answer* (Section 7.2), which we have adapted to XML retrieval. We have found that all analysed filters intersect with their underlying aboutness systems. They reflect therefore the specific XML retrieval idea to focus the underlying aboutness system's result set and not to fundamentally change it.

Overall, we have analysed three types of filters used in INEX 2005, a simple brute-force filter that keeps the highest ranked element of each XML path and two more complex filters that take into account the relations in the tree hierarchy between retrieved elements. The brute-force filter, as the most commonly used one, almost completely changes, e.g., monotonic behaviour. For XML vector space retrieval model, for instance, we could show how the brute-force filter eliminates many of its advanced reasoning capabilities.

9.1.3.3 Experimental Evaluation

In the final chapter, we wanted to demonstrate another use of theoretical evaluation. The fact that we consider actual IR models from INEX 2005, distinguishes our work from many other theoretical evaluation approaches. We compared our theoretical evaluation results with the experimental ones for XML retrieval in INEX 2005 to find out how the adjustment of existing flat document retrieval models compares to the creation of completely new ones, especially designed to meet the requirements of XML retrieval. We went through each of the three INEX 2005 XML retrieval evaluation tasks and determined reasoning properties that supported good performance for these tasks. Again, the monotonic reasoning rules have played an important role here. To our knowledge, there has not been a similar attempt to use theoretical evaluation to explain actual experimental results for XML retrieval models.

Yet, particularly in this final chapter, we could note some disadvantages that need to be discussed in relation to the strengths of our approach.

9.2 Strengths and Limitations of the Approach

In this section, we reflect on our experience with a theoretical evaluation approach. We first discuss some strengths to afterwards talk about perceived weaknesses. One remark, however, should be made from the outset. That we can easily reflect on the advantages and disadvantages of the approach is also linked to the approach itself. As we operate on a high-level of abstraction, weaknesses seem to be more apparent than in other evaluation approaches, where statistics only seemingly present a convincing abstraction of how well an IR matching function is able to describe what human users perceive as relevant.

The first advantage of a theoretical evaluation is the widened perspective. Aboutness characteristics qualify XML retrieval functions, which are represented by the number of properties they implement and they do not implement. This is certainly not as obvious for a purely statistical evaluation of a scoring function.

Logic-based evaluation is more open to debate, as underlying assumptions of IR performance can sometimes be hidden by tuning a priori assigned parameters in such a way that they fit best the evaluation task. This transparency leads to new insights about the behaviour of models in general and not only for particular evaluation tasks. If we, e.g., would like to come up with a new XML retrieval system, we should be carefully considering the degree to which we allow monotonic behaviour. The control of such behaviour could be done by defining exactly what precludes an information from being a misinformation.

These were some of perceived advantages of a logic-based theoretical evaluation. However, there are also drawbacks. Some of them might prevent researchers from further engaging with the approach. Much of XML retrieval work is currently done by adjusting weights to meet the different requirements. It has been noted [Wong et al., 2001] that the proposed theoretical evaluation formalisms often deliver too high an abstraction to cover specific cases. For XML retrieval, this will be particularly noticed when dealing with filters and when comparing experimental and theoretical evaluation results. Filters are often relatively simple mathematical operations rather than advanced reasoning. We have tried to address these problems by introducing some mathematics into the Situation Theory framework delivered by Huibers and others. As discussed in Section 4.4, to this end, we have added to his analysis the notion of conditionally supported reasoning properties.

Where we have analysed concrete XML retrieval systems, Huibers has looked at classes of different IR approaches. Heavily numeric models, however, are also difficult to represent with our methodology, as the conditions on reasoning behaviour are often not simple. In summary, we think that further research needs to be done into possible frameworks of theoretical evaluations. It might turn out that logic-based frameworks fall behind frameworks based, e.g., on retrieval heuristics. As XML retrieval is a relatively young discipline compared to traditional IR, such heuristics, however, do not seem to be established yet. Thus, we could not use them in this thesis, but this might change for the future.

To do further research into possible frameworks of theoretical evaluation, we would need a framework to evaluate theoretical evaluation frameworks, similarly to the way we have evaluated XML retrieval evaluation with our logic-based framework. Such a framework might have been suggested by [van Rijsbergen, 2004].

9.3 Further Results

Some of our conclusions are not directly the result of developing a new theoretical methodology and afterwards applying it to XML retrieval models. These conclusions often develop from discussing a particular topic of interest in detail and are specific to its research context. They are easy to miss, but are important interventions in ongoing debates.

In terms of further results, we consider one example for additional explanations with regard to experimental behaviour, one example for discussing further INEX specifics, one example about how the analysis has helped understand better the interaction of content and structure in INEX, one example of how we help improve existing models, one example of how we go beyond existing aboutness approaches and finally one example of where our analysis intervenes in general IR debates.

- Considering further explanations for experimental behaviour in INEX, while discussing the idea of an utility prior and the implied reasoning in Section 7.3.2.2, we analysed how the prior can have a strong impact on monotonic reasoning, especially if thresholds are involved. We were able to relate this to improvements in the experimental behaviour of particular models.

- Several suggestions were made how to improve existing INEX evaluation procedures. In Section 6.5.3, we could (a) explain in more detail and from an aboutness point of view why the focus on specificity from INEX 2005 onwards was correct. We could (b) derive which INEX quantisation functions focus not just on general or strict evaluation results but favour either exhaustivity or specificity judgments. Finally, we could (c) derive an alternative view on exhaustivity aboutness that would have strengthened this evaluation measure, because it would be based on the same relevant information as specificity.

- The impact of XML structure on the aboutness behaviour of INEX models has been one of the main themes in this thesis. In Chapter 5, we could progress through a series of models and starting from language modelling II and then GPX and contextualisation present how XML structure is not considered in itself but as a part of the underlying aboutness behaviour. For instance, only GPX considers CAS queries to be the standard query input. Also, the integration of XML structure by interpolating language models of neighbouring document components does not seem to improve the reasoning behaviour of language modelling II, because it is too close to the Jelinek-Mercer approach to smoothing and therefore close to standard flat document language modelling.

- Throughout Chapter 5, we discussed at several places how to improve existing models. For instance, we could discuss for contextualisation in Section 5.4.2.8 how the introduction of a threshold would have improved monotonic reasoning behaviour (especially in the contextualisation step) and would have led to a behaviour that one would expect from XML retrieval models.

- Our new theoretical evaluation framework allows us to move beyond existing theoretical evaluation approaches. For instance, our analysis of experimental behaviour was also helped by understanding why reasoning rules are not supported in an aboutness system. Cut reasoning is not supported for many of our models, because relevant elements can be cut away. In Section 8.3.3, we could identify that the Cut-induced reduction of relevant information appears less frequent in the language modelling I index which excludes small elements. This helped us explain an improved experimental performance.

- Our investigations can aid the explanations of experimental results beyond XML retrieval. For instance, during our discussion of language modelling's internal thresholds from Section 5.3, we could confirm the observation by [Manning et al., 2008] that smoothing has an influence on the aboutness behaviour and is not neutral. We have shown how important it is to ensure that the internal threshold of language modelling is the smallest possible value. In the original language modelling paper [Ponte and Croft, 1998], the collection language model of a term t is used in case the term is not found in the document. This implies the paradox that for all terms in the document that have a lower document term frequency than the collection language frequency of t their language model contribution will be lower than t. This cannot happen, if we interpolate with the collection language model, as in XML language modelling I and II, which apply the Jelinek-Mercer approach. Then, those terms that do not occur will contribute their collection language value and all those that do occur will contribute their collection and document language model. Thus, the contribution of the latter is always larger than the contribution of the former. It is therefore not surprising that the Jelinek-Mercer approach improves experimental performance of models [Manning et al., 2008].

This concludes our discussion of our results. In the final section, we offer some possible future work.

9.4 Future Work

This thesis has proposed at a theoretical evaluation of XML retrieval. Although in many parts the work might look only theoretical, most of it is characterised by the attempt to apply theory in new ways and to new problems that have emerged in recent years in IR. As a foundational work, we could only show snapshots of possible ways to progress. To this end, we have chosen only some XML retrieval models. For these, we have concentrated on some telling aboutness behaviour characteristics. Finally, we have used only some of the derived aboutness characteristics to explain experimental behaviour. Thus, there are many ways to continue the work of this thesis. It could be continued either by building on the foundations to develop new theoretical analyses (just like we have used Huibers' work) or by going into more depth with some of the existing explanations. Possible areas for future work include:

9.4.1 Enhancement of the Existing Theoretical Framework

As it is built upon proven existing theoretical frameworks, the approach presented in this thesis has already got a certain degree of maturity. Nevertheless, there remain further open questions. The most obvious one and the one we have touched upon already in our discussion of the results in the conclusion is the question how to determine exactly which aboutness rules help best with a pragmatic analysis of XML retrieval systems. We have found the analysis of monotonic reasoning rules to be particularly useful. In the future, one could further specify which rules were the most useful ones and concentrate on those. Some rules such as the conservative aboutness rules have not contributed to the analysis of XML retrieval systems because none of the analysed systems uses the more semantically oriented preclusion. We could have therefore left out these rules completely. We needed to be careful not to disregard rules prematurely. The Containment rule, for instance, has proven to be useful to understand the performance of the XML vector space model. Models also evolve. Rules that have been neglected so far could be more useful in the future. The enhancement of the existing framework should start with analysing more carefully the aboutness rules that have proven to be useful, while remaining careful not to prematurely dismiss rules.

9.4.2 Expansion of the Current Theoretical Analysis

We have mainly concentrated on mature and successful models from INEX in this thesis. We have done so in order to demonstrate the power of our theoretical analysis to also determine minute differences. It would have also been interesting to find our more about the reasons why the performance of certain models is much worth than others. We have left out the group of worst performing models altogether as we have found that they either entered only one or two years of the INEX evaluation or they reported particular problems with the implementation. As a control group to explain good *and* bad performance, however, they would have been useful. A second expansion of the current theoretical analysis could be the richer integration of Section 6.4's user models into the theoretical analysis, for which we have shown how Situation Theory can be used to express the reasoning behind specificity and exhaustivity assessments. We have only demonstrated very briefly how the reasoning of agents and systems could be brought together. Afterwards, we have concentrated on addressing specific INEX problems such as the exhaustivity evaluation dimension and filters by developing a theoretical justification. Our user models are, however, more generic and could be used to develop new theories about the performance of XML retrieval systems. Finally, throughout the thesis, we have only offered snapshots of possible ways to proceed with theoretical evaluations. We could easily expand the current theoretical analysis by systematising this approach and developing new versions of particular parts of our theoretical analysis. The analysis of filter, for instance, could benefit from a more in-depth comparison of XML retrieval filters as well as from a comparison with filters as they are used in other fields of IR.

9.4.3 Evaluation of Theoretical Frameworks

If theoretical evaluation frameworks are to expand their reach, we will need better ways to effectively evaluate them. At the beginning of our work, we did an ad-hoc examination of existing frameworks and decided to use an aboutness-based one and to use Situation Theory to express aboutness. Furthermore, we decided to take up not just one particular existing approach but to use parts of various successful models. For instance, we have enhanced Huibers' work by adding the notion of conditionally supported rules. Such decisions on the framework would benefit from a systematic investigation into best theoretical evaluation strategies. These would include the comparison of theoretical evaluation approaches and the determination of decision rules for employing various theoretical evaluation framework components. We see great potential in theoretically analysing the relevance score as a function with various variables that often include terms and their frequency values, but also other parameters. We suggest to study aboutness rules and monotonicity and how they behave with respect to these variables, but we need better ways of determining which theoretical evaluation approaches have been successful in describing this aboutness behaviour.

9.4.4 Integration of Experimental Evaluation

We had decided early to concentrate on a theoretical evaluation. As we covered a new field for theoretical evaluation with the analysis of XML retrieval systems, we first had to develop a methodology for the theoretical evaluation and afterwards show that this methodology covers the important research areas in XML retrieval. This meant that we decided not to include experimental evaluation in our work, although at some points during our analysis we were able to give concrete recommendations for the improvement of XML retrieval models. It would have been useful to verify these suggestions by including an experimental evaluation alongside our theoretical evaluation and thus prove that a theoretical evaluation is useful to understand and improve existing systems. Integrating experimental evaluation will be the focus of our immediate follow-on work. This should also help convince a larger IR community of the usefulness of our aboutness approach.

Finally, a closer tie to experimental evaluation will lead to the ability to theoretically think through a model during its design phase. Throughout the thesis, we have made various suggestions on how to improve existing models. Some of these can be generalised and should help with the development of new models. These new models would be theoretically sound and would show an expected experimental behaviour by using the insights from the theoretical pre-evaluation during the model design phase.

9.4.5 New Application Areas

One of the advantages of the approach presented here is that it helps an emerging field before it is mature enough to develop its own evaluation strategies that reflect its specific requirements. XML retrieval is by now very mature and has its own evaluation regime with INEX. Other fields are not as mature or not yet big enough to include dedicated evaluation strategies. The presented methodology can help in the early stages to structure design approaches and develop evaluation strategies. Of particular interest will be in the near future to develop new information retrieval strategies for the emerging web of things, i.e. a web where devices and objects are directly interlinked. Because the web of things relies on a graph-based data model using the W3C standard RDF, our approach, which combines structure and content, could be useful and easily adopted. In fact, there are many other emerging fields in information retrieval, which use evidence from a network of information to enhance the retrieval process. These include opinion mining or expert systems, which both use networks of related information (reviews, expert assessments, etc.), to return relevant results. Here, our approach could help with design decisions for emerging models but also help to understand how traditional information retrieval techniques could be reused for the new approaches.

9.5 Declaration

I herewith declare that I have produced this thesis without the prohibited assistance of third parties and without making use of aids other than those specified; notions taken over directly or indirectly from other sources have been identified as such. This thesis has not previously been presented in identical or similar form to any other national or foreign examination board.

LONDON, 30/8/2011
Tobias Blanke

References

Paavo Arvola, Marko Junkkari, and Jaana Kekäläinen. Generalized contextualization method for XML information retrieval. In *ACM CIKM '05*, New York, NY, USA, 2005a. 120

Paavo Arvola, Jaana Kekäläinen, and Marko Junkkari. Query evaluation with structural indices. In *INEX*, pages 134–145, 2005b. 120, 121, 122, 182, 186

Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999. 8, 17, 54

Jon Barwise and John Etchemendy. *Language, Proof and Logic*. Center for the Study of Language and Inf, 2002. 49

Jon Barwise and John Perry. *Situations and Attitudes*. Cambridge University Press, MA, 1983. 23

Jon Barwise and Jerry Seligman. *Information Flow: The Logic of Distributed Systems*. Cambridge University Press, Cambridge, 1997. 26, 27

Tobias Blanke and Mounia Lalmas. Theoretical benchmarks of XML retrieval. In *ACM SIGIR '06*, pages 613–614, New York, NY, USA, 2006. ACM Press. 140

Allen L. Brown, Jr., Surya Mantha, and Toshiro Wakayama. Preference logics and non-monotonicity in logic programming. In *TVER '92: Proceedings of the Second International Symposium on Logical Foundations of Computer Science*, pages 45–56, London, UK, 1992. 45

Hans Henrik Brunn. *Science, Values and Politics in Max Weber's Methodology*. Ashgate, London, January 2007. 60, 82

Peter Bruza and Theo W. C. Huibers. A study of aboutness in information retrieval. *Artif. Intell. Rev.*, 10(5-6):381–407, 1996. 35

Peter D. Bruza and Theo W. C. Huibers. Investigating aboutness axioms using information fields. In *ACM SIGIR '94*, pages 112–121, New York, NY, USA, 1994. 22, 41

Yves Chiaramella. Information retrieval and structured documents. In *Lectures on information retrieval*, pages 286–309. New York, 2001. 29, 30, 32, 33, 34, 35, 42, 138, 146, 183, 189

Yves Chiaramella and Jean P. Chevallet. About retrieval models and logic. *Comput. J.*, 35(3):233–242, 1992. 19

Charles L. A. Clarke. Controlling overlap in content-oriented XML retrieval. In *ACM SIGIR '05*, pages 314–321, New York, NY, USA, 2005. 160, 161, 162

William S. Cooper. A definition of relevance for information retrieval. *Information Storage and Retrieval*, 7:19–37, 1971. 19

Keith Devlin. *Logic and information*. Cambridge University Press, New York, NY, USA, 1991. 23, 24, 25, 26, 42

Keith Devlin. Situation theory and social structure. In *International Conference Logic at Work on Knowledge Representation and Reasoning Under Uncertainty, Logic at Work*, pages 197–237, London, UK, 1994. 24, 25, 28, 60

Keith Devlin. Modeling real reasoning. In *Formal Theories of Information: From Shannon to Semantic Information Theory and General Concepts of Information*, pages 234–252. Springer-Verlag, 2009. 27

Fred Dretske. *Knowledge and the Flow of Information*. MIT Press, Cambridge, MA, 1981. 24, 27, 49

Hui Fang and ChengXiang Zhai. An exploration of axiomatic approaches to information retrieval. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 480–487, New York, NY, USA, 2005. 21

Hui Fang, Tao Tao, and ChengXiang Zhai. A formal study of information retrieval heuristics. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 49–56, New York, NY, USA, 2004. 21, 186

Norbert Fuhr, Mounia Lalmas, Saadia Malik, and Gabriella Kazai, editors. *Advances in XML Information Retrieval and Evaluation, INEX 2005, Dagstuhl*, volume 3977 of *Lecture Notes in Computer Science*, 2006. 166, 203, 204, 205

Shlomo Geva. GPX - Gardens point XML IR at INEX 2005. In Fuhr et al. [2006], pages 240–253. 110, 113, 114, 117, 181, 185

Norbert Gövert, Gabriella Kazai, Norbert Fuhr, and Mounia Lalmas. Evaluating the effectiveness of content-oriented XML retrieval. *Journal of Information Retrieval*, 9(6): 699–722, 2006. 13, 14, 32, 61, 84, 132, 133, 135, 136, 143, 147, 183

David A. Grossman and Ophir Frieder. *Information Retrieval: Algorithms and Heuristics*. Springer, Dordrecht, 2. edition, 2004. 42

Theo W. Huibers, Iadh Ounis, and Jean-Pierre Chevallet. Conceptual graph aboutness. In *Proceedings of the 4th International Conference on Conceptual Structures: Knowledge Representation as Interlingua*, pages 130–144, London, UK, 1996a. 55, 62, 63, 64, 66, 68

Theo W. C. Huibers, Mounia Lalmas, and Keith J. van Rijsbergen. Information retrieval and situation theory. *SIGIR Forum*, 30(1):11–25, 1996b. 40

Theo W.C. Huibers. *An Axiomatic Theory for Information Retrieval*. Universiteit Utrecht, Utrecht, 1996. 17, 18, 19, 20, 21, 22, 24, 26, 29, 31, 36, 37, 39, 40, 41, 42, 43, 44, 45, 46, 47, 50, 51, 52, 53, 55, 56, 60, 67, 68, 69, 74, 82, 86, 93, 99, 115, 135, 136, 137, 151, 152

Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002. 132, 174

Gabriella Kazai and Mounia Lalmas. INEX 2005 evaluation measures. In Fuhr et al. [2006], pages 16–29. 12, 131, 146, 151, 166, 167, 168, 172, 175, 179, 181

Gabriella Kazai and Mounia Lalmas. Extended cumulated gain measures for the evaluation of content-oriented XML retrieval. *ACM Trans. Inf. Syst.*, 24(4):503–542, 2006. 132

Gabriella Kazai and Mounia Lalmas. Notes on what to measure in INEX. In *INEX 2005 Workshop on Element Retrieval Methodology*, Glasgow, July 2005. 132, 133

Sarit Kraus, Daniel Lehmann, and Menachem Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artif. Intell.*, 44(1-2):167–207, 1990. 43

Mounia Lalmas. *Theories of Information and Uncertainty for the modelling of Information Retrieval: an application of Situation Theory and Dempster-Shafer's Theory of Evidence*. PhD thesis, University of Glasgow, Department of Computing Science, 1996. 24, 25

Mounia Lalmas. Dempster-shafer's theory of evidence applied to structured documents: modelling uncertainty. In *SIGIR '97: Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 110–118, New York, NY, USA, 1997. 24

Mounia Lalmas. XML retrieval. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 1(1):1–111, 2009. 7, 11, 12, 13, 14, 22, 64

Mounia Lalmas and Ricardo Baeza-Yates. Structured text retrieval. In *Modern Information Retrieval*. Springer-Verlag, New York, 2010. 7, 8

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. 7, 8, 9, 11, 12, 23, 54, 90, 91, 92, 196, 197

Yosi Mass and Matan Mandelbrod. Using the INEX environment as a test bed for various user models for XML retrieval. In Fuhr et al. [2006], pages 187–195. 76, 77, 80, 163, 176, 177, 184

Vojkan Mihajlovic, Georgina Ramírez, Thijs Westerveld, Djoerd Hiemstra, Henk Ernst Blok, and Arjen P. de Vries. Tijah scratches INEX 2005: Vague element selection, image search, overlap, and relevance feedback. In Fuhr et al. [2006], pages 72–87. 153, 156, 157, 158, 159

Elke Mittendorf and Peter Schäuble. Document and passage retrieval based on hidden markov models. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 318–327, New York, NY, USA, 1994. 30

Jian-Yun Nie. An outline of a general model for information retrieval systems. In *SIGIR '88: Proceedings of the 11th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 495–506, New York, NY, USA, 1988. 33, 35, 189

Jian Yun Nie. Towards a probabilistic modal logic for semantic-based information retrieval. In *In Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 140–151, 1992. 20

Paul Ogilvie and Jamie Callan. Hierarchical language models for XML component retrieval. In Norbert Fuhr, Mounia Lalmas, Saadia Malik, and Zoltán Szlávik, editors, *INEX*, volume 3493 of *Lecture Notes in Computer Science*, pages 224–237, 2004. 102

Paul Ogilvie and Jamie Callan. Parameter estimation for a simple hierarchical generative model for XML retrieval. In Fuhr et al. [2006], pages 211–224. 90, 102, 104, 107

Paul Ogilvie and Mounia Lalmas. Investigating the exhaustivity dimension in content-oriented XML element retrieval evaluation. In *15th ACM Conference on Information and Knowledge Management (CIKM 2006)*, Arlington, VA, 2006. 136, 140, 142, 144, 145, 146

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. In *Proceedings of the 7th International World Wide Web Conference*, pages 161–172, Brisbane, Australia, 1998. 31

J. Pehcevski and J. A. Thom. Hixeval: Highlighting XML retrieval evaluation. In *Advances in XML Information Retrieval and Evaluation: Fourth Workshop of the INitiative for the Evaluation of XML Retrieval (INEX 2005). Dagstuhl 28-30 November 2005*, pages 43–57. New York, 2006. 142

Benjamin Piwowarski and Mounia Lalmas. Structured information retrieval and quantum theory. In *3rd Quantum Interaction Symposium, DFKI, Saarbruecken*, 2009. 18

Jay M. Ponte and W. Bruce Croft. A language modelling approach to information retrieval. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281, New York, NY, USA, 1998. 90, 106, 197

Stephen E. Robertson, Steve Walker, Micheline Hancock-Beaulieu, Aarron Gull, and Marianna Lau. Okapi at TREC-3. In *Text REtrieval Conference*, pages 21–30, 1992. 120

Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975. 54

Fabrizio Sebastiani. On the role of logic in information retrieval. In *Information Processing and Management*, pages 1–18, 1998. 19, 21

Börkur Sigurbjörnsson and Jaap Kamps. The effect of structured queries and selective indexing on XML retrieval. In Fuhr et al. [2006], pages 104–118. 90, 94, 97, 102, 163, 171, 172, 178

David Song and Peter D. Bruza. Towards context sensitive information inference. *Journal of the American Society for Information Science and Technology (JASIST)*, 54:321–334, 2003. 27, 28

John F. Sowa. Conceptual graphs. *Knowl.-Based Syst.*, 5(3):171–172, 1992. 62

Andrew Trotman. Wanted: Element retrieval users. In *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology*, pages 58–64, Glasgow, 2005. 142

Andrew Trotman and Boerkur Sigurbjoernsson. Narrowed extended XPath (NEXI). In *In Proceedings of the INEX 2004 Workshop*, pages 16–40. Springer-Verlag, 2004. 14

Howard Turtle and W. Bruce Croft. Evaluation of an inference network-based retrieval model. *ACM Trans. Inf. Syst.*, 9(3):187–222, 1991. 20

Keith J. van Rijsbergen. *Information retrieval.* Butterworths, London, UK, 2 edition, 1979. 7, 8

Keith J. Van Rijsbergen. A non-classical logic for information retrieval. *The Computer Journal*, 29(6):481–485, 1986a. 19, 20, 51

Keith J. Van Rijsbergen. A new theoretical framework for information retrieval. *SIGIR Forum*, 21:23–29, 1986b. 19

Keith J. van Rijsbergen. Towards an information logic. In *Proceedings of the 12th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '89, pages 77–86, New York, NY, USA, 1989. 18

Keith J. van Rijsbergen. Another look at the logical uncertainty principle. *Inf. Retr.*, 2 (1):17–26, 2000. 35, 90

Keith J. van Rijsbergen. *The Geometry of Information Retrieval.* Cambridge University Press, 2004. 23, 36, 40, 54, 133, 195

Keith J. van Rijsbergen and Mounia Lalmas. Information calculus for information retrieval. *Journal of the American Society for Information Science and Technology (JASIST),*, 47(5):385–398, 1996. 22

Ellen M. Voorhees and Donna K. Harman. *TREC: Experiment and Evaluation in Information Retrieval.* Digital Libraries and Electronic Publishing. 2005. 13, 17

Max Weber. *The methodology of the social sciences.* Free Press, New York, NY, USA, 1997 (1903-1917). 60

Dominic Widdows. Orthogonal negation in vector spaces for modelling word-meanings and document retrieval. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 136–143, Morristown, NJ, USA, 2003. 50

Ludwig Wittgenstein. *Tractatus Logico-Philosophicus.* Suhrkamp Verlag, Frankfurt/Main, 1922. 24, 29

Kam-Fai Wong, Dawei Song, Peter Bruza, and Chun-Hung Cheng. Application of aboutness to functional benchmarking in information retrieval. *ACM Trans. Inf. Syst.*, 19(4): 337–370, 2001. 20, 22, 28, 39, 41, 43, 44, 46, 48, 51, 52, 54, 55, 56, 58, 59, 74, 82, 84, 178, 195

Song K. M. Wong and Yao Y. Yao. On modelling information retrieval with probabilistic inference. *ACM Trans. Inf. Syst.*, 13(1):38–68, 1995. 133