Mixture Models, Overlap, and Structural Hints in XML Element Retrieval

Börkur Sigurbjörnsson[1], Jaap Kamps[1,2], and Maarten de Rijke[1]

[1] Informatics Institute, University of Amsterdam
[2] Archives and Information Studies, Faculty of Humanities, University of Amsterdam

Abstract. We describe the INEX 2004 participation of the Informatics Institute of the University of Amsterdam. We completely revamped our XML retrieval system, now implemented as a mixture language model on top of a standard search engine. To speed up structural reasoning, we indexed the collection's structure in a separate database. Our main findings are as follows. First, we show that blind feedback improves retrieval effectiveness, but increases overlap. Second, we see that removing overlap from the result set decreases retrieval effectiveness for all metrics except the XML cumulative gain measure. Third, we show that ignoring the structural constraints gives good results if measured in terms of mean average precision; the structural constraints are, however, useful for achieving high initial precision. Finally, we provide a detailed analysis of the characteristics of one of our runs. Based on this analysis we argue that a more explicit definition of the INEX retrieval tasks is needed.

1 Introduction

We follow an Information Retrieval (IR) approach to the Content-Only (CO) and Vague-Content-And-Structure (VCAS) ad hoc tasks at INEX. In our participation at INEX 2004 we built on top of our element-based approach at INEX 2003 [10], and extended our language modeling approach to XML retrieval.

Specifically, we addressed the following technological issues, mainly to obtain a statistically more transparent approach. For our INEX 2003 experiments we combined article and element scores outside our language modeling framework. That is, we calculated scores separately for articles and elements, and then updated the element scores by taking into account the score of the surrounding article. This year, we implemented a proper mixture language model, incorporating evidence from both the XML elements and the article in which they occur. Furthermore, at INEX 2003 we estimated the language model of the collection by looking at statistics from our overlapping element index. For our INEX 2004 experiments we estimate this collection model differently, by looking at statistics from our article index. The main change in our blind feedback approach, compared to last year, is that this year we perform query expansion based on an element run, whereas last year we performed the expansion based on an article run. All our runs were created using the ILPS extension to the Lucene search engine [7, 3].

Our main research questions for both tasks were twofold. First, we wanted to investigate the effect of blind feedback on XML element retrieval. Second, we wanted to cast light on the problem of overlapping results; in particular, we investigate the effect of removing overlapping results top-down from a retrieval run. A third, additional research question only concerns the VCAS task: we investigate the difference between applying a content-only approach and a strict content-and-structure approach.

The remainder of this paper is organized as follows. In Section 2 we describe our experimental setup, and in Section 3 we provide details on the official INEX 2004 runs. Section 4 presents the results of our experiments, and in Section 5 we analyze the characteristics of one of our runs. In Section 6 we conclude.

2 Experimental Setup

2.1 Index

Our approach to XML retrieval is IR-based. We calculate a retrieval score for an element by using a mixture model which combines a language model for the element itself, a model for the surrounding article, and a model for the whole collection. To estimate the language models we need two types of inverted indexes, one for the XML elements and another for the full XML articles. The two indexes have a somewhat different purpose. The element index functions as a traditional inverted index used to retrieve elements. The article index, on the other hand, is used for statistical estimates only. Furthermore, we maintain a separate index of the collection structure.

Element index We index each XML element separately. The indexing unit can be any XML element, ranging from small elements such as words in italics (⟨`it`⟩) to full blown articles (⟨`article`⟩). For each element, all text nested inside it is indexed. Hence, the indexing units overlap (see Figure 1). Text appearing in a particular nested XML element is not only indexed as part of that element, but also as part of all its ancestor elements.

Article index Here, the indexing unit is a complete XML document containing all the terms appearing at any nesting level within the ⟨`article`⟩ tag. Hence, this is a standard inverted index as used for traditional document retrieval.

Both the element and the article index were word-based: we applied case-folding, and stop-words were removed using the stop-word list that comes with the English version on the Snowball stemmer [12], but other than that words were indexed as they occur in the text, and no stemming was applied.

Structure index The structure of the collection is indexed using a relational database. To index the XML trees we use pre-order and post-order information of the nodes in the XML trees [1].

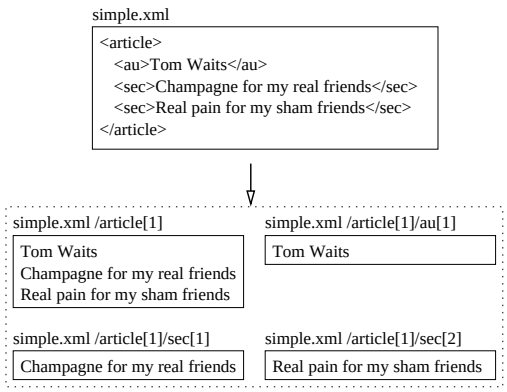

Fig. 1. Figure of how XML documents are split up into overlapping indexing units.

2.2 Query Processing

For both the CO and the VCAS task we only use the ⟨`title`⟩ part of the topics. We remove words and phrases bounded by a minus-sign from the queries; other than that, we do not use the plus-signs, or phrase marking of the queries.

For the CAS topics we have a NEXI tokenizer which can decompose the query into a set of `about` functions [11]. If there is a disjunction in a location-path, we break it up into a disjunction of about functions. That is,

$$\texttt{about(.//(abs|kwd), xml)} \tag{1}$$

becomes

$$\texttt{about(.//abs,xml) or about(.//kwd,xml).} \tag{2}$$

If there are multiple about functions with the same scope we merge them into a single one. That is,

$$\texttt{about(., broadband) or about(., dial-up)} \tag{3}$$

becomes

$$\texttt{about(., broadband dial-up).} \tag{4}$$

For some of the VCAS runs we ignore the structural constraints and use only a collection of content query terms. That is, from the query

$$\texttt{//article[about(.,sorting)]//sec[about(.,heap sort)]} \tag{5}$$

we collect the query terms

$$\texttt{sorting heap sort.} \tag{6}$$

We will refer to these as the *full content queries*.

2.3 Retrieval Model

All our runs use a multinomial language model with Jelinek-Mercer smoothing [2]. We estimate a language model for each of the elements. The elements are then ranked according to their prior probability of being relevant and the likelihood of the query, given the estimated language model for the element:

$$P(e|q) \propto P(e) \cdot P(q|e). \tag{7}$$

We assume query terms to be independent, and rank elements according to:

$$P(e|q) \propto P(e) \cdot \prod_{i=1}^{k} P(t_i|e), \tag{8}$$

where q is a query made out of the terms $t_1, \ldots, t_k$. To account for data sparseness we estimate the element language model by taking a linear interpolation of three language models: one for the element itself, one for the article that contains the element, and a third one for the collection. That is, $P(t_i|e)$ is calculated as

$$\lambda_e \cdot P_{mle}(t_i|e) + \lambda_d \cdot P_{mle}(t_i|d) + (1 - \lambda_e - \lambda_d) \cdot P_{mle}(t_i), \tag{9}$$

where $P_{mle}(\cdot|e)$ is a language model for element e; $P_{mle}(\cdot|d)$ is a language model for document d; and $P_{mle}(\cdot)$ is a language model of the collection. The parameters λ_e and λ_d are interpolation factors (smoothing parameters). We estimate the language models, $P_{mle}(\cdot|\cdot)$ and $P_{mle}(\cdot)$, using maximum likelihood estimation. For the element model we use statistics from the element index; for the document model we use statistics from the article index; and for the collection model we use document frequencies from the article index.

The language modeling framework allows us to easily model non-content features. One of the non-content features that proved to be useful during our experiments for INEX 2003 is document length [4]. Specifically, we assign a prior probability to an element e relative to its length in the following manner:

$$P(e) = \frac{|e|}{\sum_e |e|}, \tag{10}$$

where $|e|$ is the size of an element e.

2.4 Query Expansion

We have been experimenting with blind feedback in every edition of INEX so far, focusing on query expansion for the content-only task exclusively. Initially, we experimented with Rocchio-style reweighting to select up to 10 terms from the top 10 documents [9]. In INEX 2002 we observed that query expansion with Rocchio on the article index gave intuitively useful expanded queries, leading to the kind of improvements that are familiar from article retrieval [5]. However, expanding queries based on the top 10 retrieved XML elements seemed not to

work due to the short and overlapping elements in the top 10 results. Hence, we decided to expand queries on the article index, and then run the expanded queries against the element index. This did, indeed, give us a boost for the 2002 topics, but, alas, substantially lowered our score for the 2003 topics [10].

Our analysis of the failure of article-index based feedback in INEX 2003 was that the terms were useful, but highly unlikely to occur in the proper element. An example is getting prominent author names from the bibliography, which are relevant and useful retrieval cues but generally do not appear in a paragraph (maybe in the author field, or the bibliography).[3]

We decided to go back to the idea of doing blind feedback on the XML element index. This has the advantage of conservatism, the initially retrieved top 10 elements will keep their high ranking, but the problem of overlap in the initial result set remains. In pre-submission experiments, Ponte's language modeling approach to feedback [8] proved more robust, and improved performance on the 2003 topics.

3 Runs

In this section we describe our runs submitted for INEX 2004. All our runs use the language modeling framework described in the previous section. For all runs we use a two level smoothing procedure: we smooth against both the article and the collection. Our collection model uses the document frequencies from the article index. For computing the likelihood of a term given an element, see Equation 9, we use the following parameter settings for all runs: $\lambda_e = 0.1$ and $\lambda_d = 0.3$. All runs also use the same length prior settings in Equation 10. All the runs that use blind feedback use the language modeling approach [8], and consider the top 15 elements as pseudo relevant, and expand the query with up to 5 additional terms.

3.1 Content-Only Runs

Baseline (UAms-CO-T) This run uses the mixture language model approach and parameter settings as described above.

Feedback (UAms-CO-T-FBack) This run uses the same model and parameters as the previous run. Additionally, it uses blind feedback to expand the queries, as described above.

No Overlap (UAms-CO-T-FBack-NoOverl) This run uses the same model, parameters and feedback approach as the previous run. Additionally, overlapping results are filtered away. The filtering is done in a top-down manner. That is, the result list is processed from the most relevant to the least relevant element.

[3] We have been planning to incorporate context (i.e., tags in which term occurs) into our model, but this would requires some CAS features for the CO runs that are non-trivial to implement.

Table 1. Average scores for our submissions, with the best scoring run in italics. The first two rows (MAP, P@10) are results of using `trec_eval` and strict assessments. The next three rows are results of using `inex_eval`. Finally, we list the XCG scores that have been released and the set-based overlap in the runs. (Left): Our CO runs. (Right): Our VCAS runs.

	CO-Runs			VCAS-Runs		
Measure	Baseline	Feedback	No Overlap	CO-style	XPath-style	No Overlap
MAP	0.1142	*0.1216*	0.0385	*0.1377*	0.0811	0.0621
P@10	*0.1680*	0.1600	0.1000	0.1818	*0.2591*	0.1121
strict	0.1013	*0.1100*	0.0332	*0.1260*	0.0735	0.0582
generalized	0.0929	*0.1225*	0.0198	*0.1167*	0.0451	0.0330
so	0.0717	*0.1060*	0.0149	*0.0912*	0.0472	0.0282
XCG	–	0.2636	*0.3521*	–	–	–
Overlap	72.0%	81.8%	0.0%	77.8%	18.8%	0.0%

A result is removed from the result list if its content overlaps with the content of an element that has been processed previously.

3.2 Vague Content-And-Structure Runs

CO-style (UAms-CAS-T-FBack) This run uses the full-content version of the queries. The run is identical to CO-Feedback, except for the topics, of course.

No Overlap (UAms-CAS-T-FBack-NoOverl) This run uses the full-content version of the queries. The run is identical to CO-No-Overlap.

XPath-style (UAms-CAS-T-XPath) This run is created using our system for the INEX 2003 Strict Content and Structure task. It uses both content and structural constraints. Target constraints are interpreted as strict. We refer to [11] for a detailed description of the retrieval approach used. The run uses the exact same approach and settings as the run referred to as "Full propagation run" in that paper.

4 Results and Discussion

In this section we present the results of our retrieval efforts. Result analysis for XML retrieval remains a difficult task: there are still many open questions regarding how to evaluate XML element retrieval. A plethora of measures has been proposed, but still the problem has not been resolved. We will present results for a number of the suggested measures and try to interpret the flow of numbers. All results are based on version 3.0 of the assessments.

4.1 Content-Only Task

For the content-only task we focus on two issues, effect of blind feedback and overlap removal.

Table 2. Tag-name distribution of retrieved elements. We only list the most frequently occurring elements

Tag-name	Baseline	Feedback	No Overlap
article	27.2%	14.3%	46.1%
bdy	21.2%	12.3%	5.7%
sec	16.6%	16.9%	7.1%
p	7.8%	17.2%	12.8%

Blind feedback From Table 1 (Left) we see that the run which uses blind feedback outperforms the normal run on all metrics except for early precision. Note also that the overlap in the feedback run is somewhat higher than for the baseline. Unfortunately, we do not have the XCG score for our baseline run. It is thus not clear how the less overlapping baseline compares to the feedback run for that measure.

The changing overlap percentage is just one of a larger set of changes brought about by applying blind feedback. Table 2 shows the distribution of retrieved elements over the most common elements. We see that by applying blind feedback, our retrieval focus changes from mostly retrieving articles and bodies, to retrieving more sections and paragraphs. We have thought of two possible explanations for this behavior. First, our retrieval model is sensitive to query length. The effect of our normal length prior decreases as the queries get longer.[4] When we expand our queries we make them longer and consequently decrease the effect of the length prior. Second, the increased overlapping might be caused by the fact that we use pseudo-relevant elements to calculate feedback terms. The goal of the added feedback terms is to attract similar content as in the pseudo-relevant elements. Obviously, the elements that overlap with the pseudo-relevant elements have to some extent very similar content.

Overlap removal Let's now take a look at a more striking difference in overlap. Namely for our feedback run with and without the list-based removal of overlapping elements. We go from having more than 80% overlap to having none at all. The removal of overlap is appreciated by neither of the two traditional metrics `trec_eval` or `inex_eval`. The XML Cumulative Gain (XCG) [6] measure is however clearly sympathetic to the removal of overlap. It is interesting to note that from Table 2 we see that almost half of the elements in our non-overlapping run are full articles.

It is tricky to draw strong conclusions from these observations. The `_eval` measures on the one hand and XCG on the other, seem to evaluate different tasks. The `_eval` measures evaluate a "system-oriented" task where the main goal is to find relevance and overlap is (and should be) rewarded. The XCG measure on the other hand evaluates a "user-oriented" task where overlap is

[4] Previously, we have shown that a more radical length bias is essential to achieve good results [4]. Those experiments were performed using both the title and description fields of the topics.

not (and should not be) rewarded. We will return to this point at the end of Section 5.

4.2 Vague Content-And-Structure task

For the VCAS task we focus on the difference between two styles of retrieval: CO and XPath. Table 1 (right) shows the results for our VCAS runs, using the different metrics. We see that the CO-style run clearly outperforms the XPath-style run with respect to all the MAP-oriented metrics. When we look at early precision, however, we see a clear distinction in favor of the XPath-style run. This finding is very much in-line with our intuition: the main reason behind adding structural constraints to content queries is to make them more precise.

The CAS topics can be divided into classes based on the extent to which structure is being used. We define 4 classes.

Restricted Search This category has topics in which structure is only used as a granularity constraint. The topic is an ordinary content-only topic, where the search is restricted to particular XML elements. There is a filter on the target element having no nested path constraint. A typical example of such a topic is to restrict the search to sections:

```
//sec[about(., ‘‘xxx’’)].
```

Contextual Content Information This category is similar to the *Restricted Search* category, but now there is a (simple) filter on the path constraint. I.e., there is a content restriction on the environment in which the requested element occurs. A typical example of such a topic is to have a content restriction on the whole article in which the requested sections occur, this may look like:

```
//article[about(., ’’xxx’’)]//sec[about(., ‘‘yyy’’)].
```

Search Hints This category contains topics with a complex filter in which a nested path occurs, but the element targeted by the nested paths resides inside the requested element. I.e., the user provides a particular retrieval cue to the system. A typical example of such a topic may be, when interested in a whole article on a topic, to tell the system to look for certain terms to appear in the abstract, this may look like:

```
//article[about(., ‘‘xxx’’) and about(.//abs, ‘‘yyy’’)].
```

Twig Hints The fourth and last category deals with topics with a nested path that targets elements that are disjoint from the requested element. This is called a tree pattern query or a ‘twig.’ Here, the user is really exploiting her knowledge of the structure of the documents, and conditions the retrieval of elements on the content found along other paths in the document tree. I.e., the condition is evaluated against parts of the text that are not being returned to the user as a result. E.g., the similar retrieval cue on the abstract, may still make sense for a user looking for sections, which may look like:

```
//article[about(.//abs, ‘‘xxx’’)]//sec[about(., ‘‘yyy’’)].
```

Table 3. Scores for different categories of CAS topics. Scores are calculated using `trec_eval` and the strict assessments. The number of assessed topics per class is stated in brackets.

	Restricted Search (5)		Contextual Content (5)		Search Hints (4)		Twig Hints (8)	
	MAP	P@10	MAP	P@10	MAP	P@10	MAP	P@10
CO	*0.1454*	0.2000	*0.1009*	0.1000	*0.3231*	0.2250	0.0631	0.2000
XPath	0.1254	*0.3000*	0.0319	*0.1200*	0.0504	*0.3500*	*0.0994*	*0.2750*

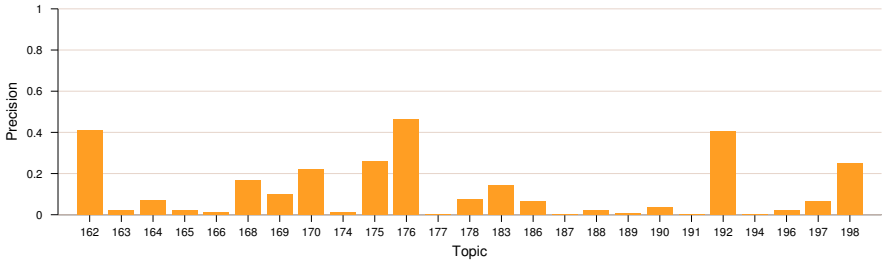

Fig. 2. Precision of our baseline for individual topics

Table 3 shows the evaluation results for different topic classes. We can see that the XPath-style approach gives better early precision for all the topic classes. Only for the most complex class of topics gives XPath-style a better MAP than CO-style approach. It is interesting to note that, in terms of MAP, the two approaches are competitive for both the simplest and most complex topics. For the two middle-classes, however, the MAP scores for the CO-style run are superior by a margin.

5 Per-topic Analysis

In this section we look at the per-topic results for our content-only baseline. Throughout this section, we will try to come up with necessary and sufficient conditions for successful XML element retrieval. Our main aim is to try to understand better how our XML retrieval system works. All experiments in this section are performed using the `trec_eval` program and the strict version of the assessments.

Figure 2 shows the precision scores for the individual topics. It is striking that for most of the topics, our performance is very poor. We score reasonably for only a handful of topics. The first questions that arise are: 'What is going wrong?' and 'What is going right?' The answers are far from obvious, since there are so many possibilities. We start by considering the core task of XML retrieval, namely, finding the relevant elements. This seems to be a combination of two problems. On the one hand we need to find relevance. On the other, we need to

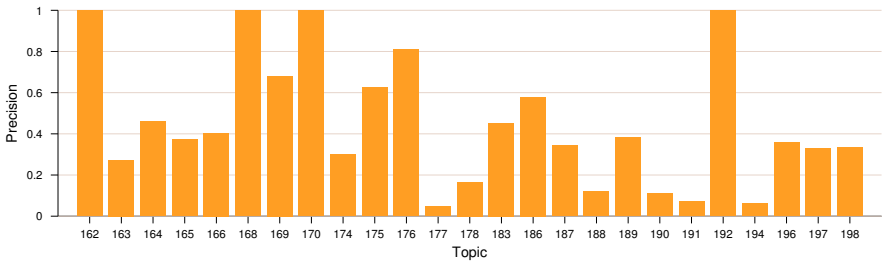

Fig. 3. Precision of document retrieval based on element score.

find the appropriate granularity of results. We will thus start with an obvious, or at least intuitive, hypothesis

– Our difficulty is more with finding the right unit, than finding the relevance

We investigate this hypothesis by trying to take the granularity problem out of the equation and see what happens. We create an article retrieval run, based on our official element run. We simply assign to an article, the score of its highest scoring element. Similarly, we define a new set of article assessments from the *strict* element assessments. A document is considered relevant if, and only if, it contains an element which is both highly exhaustive and highly specific. Figure 3 shows the precision of our new document run. In comparison to Figure 2, Figure 3 looks much more promising.

Of course, one important difference between the two evaluations is the recall base. In the case of elements, there are millions of elements that can possibly be retrieved. In the case of documents, however, the recall base "only" consists of thousands of documents. While XML element retrieval is a "needle in the haystack problem," XML document retrieval is merely a "pencil in the drawer problem." Hence, the score in the two figures are not at all comparable.

What we can do, however, is to look at topics where we did not find the appropriate elements and see if we are at least finding elements in the right documents. If we look at the recall of our element run, retrieving up to 1,500 elements per topic, we see that we find 903 out of 2,589 relevant elements (35%). The recall of the document run tells us that we find elements from 273 out of 279 articles which contain an element which is highly specific and highly exhaustive (98%). This is a clear indication that, while we may not be finding the right elements, we are at least in the neighborhood.

The differences in scoring from one topic to another may come as no great surprise. At the INEX 2004 workshop, it was reported that assessor agreement at the element level is extremely low for the strict assessments. This would imply that it is difficult to get much higher retrieval scores, unless some measures are taken to try to increase assessor agreement. The same study revealed that, even using a liberal measure of agreement at the document level, the agreement was

Table 4. The count of topics that belongs to each class of the classification of topics based on retrieval score

Class	Interval	Element-run
M_0	$0.0 \leq x < 0.05$	12
M_1	$0.05 \leq x < 0.1$	4
M_2	$0.1 \leq x < 0.2$	3
M_3	$0.2 \leq x < 0.3$	3
M_5	$0.4 \leq x < 0.5$	3

Table 5. Relation between retrieval score and the number of elements assessed as relevant. (Left): Assessment statistics for different topic classes. (Right): Topics classified by the number of relevant elements (vertical). The number of topics in each class is shown in brackets. For each class we look at the distribution over score classes (horizontal)

	# assessments			
	avg	median	min	max
M_0	125.6	33.5	4	848
M_1	212.5	128.0	13	581
M_2	58.3	10.0	2	163
M_3	8.3	4.0	3	18
M_5	10.7	10.0	5	17

	score classes (%)				
# rel.	M_0	M_1	M_2	M_3	M_5
1-10 (7)	14.3	–	28.6	28.6	28.6
11-20 (6)	50.0	16.7	–	16.7	16.7
21-40 (4)	100.0	–	–	–	–
41-80 (2)	50.0	50.0	–	–	–
> 80 (6)	50.0	33.3	16.7	–	–

still no more than 20%. Viewed in this light, the document retrieval scores of Figure 3 are surprisingly high.

Let's now look in more detail at our element retrieval run. We would like to understand better the scoring behavior of our system. When do we succeed? When do we fail? We try to analyze this by looking at the following hypotheses:

- Our system scores well if, and only if, few elements are assessed relevant;
- Our system scores well if, and only if, the assessor likes somewhat large elements;
- Our system scores well if, and only if, the recall base of the topic is overlapping.

In order to test these hypotheses we divide the topics into 5 classes, based on the precision of our baseline element retrieval run for the individual topics. Table 4 shows the topic classification. Our hypotheses so-far only consider the relation between assessments and scores. The more interesting, and more challenging, task is to relate queries and scores: this remains as future work.

Let's look at our first hypothesis: *Our system scores well if, and only if, few elements are assessed relevant.* Table 5 shows the relation between assessment count and retrieval score of our baseline system. The left table shows average, median, minimum and maximum number of relevant elements for each of our score classes. This table supports one direction of the hypothesis. That is, for

Table 6. Relation between size bias in assessments and retrieval effectiveness. (Left): The first column defines the topic classes. Then, for each class of topics, the numbers in columns 2–6 represent the percentage of assessed elements having the tag-name in the column heads. (Right): We look at sets of topics having at least 10% of their assessments of a particular tag-name (vertical). Number of topics is in brackets. For each set we look at the distribution over score classes (horizontal)

	tag-names (%)				
	article	bdy	sec	ss1	Σ
M_0	2.3	2.0	5.8	5.3	15
M_1	4.2	3.9	15.2	9.3	33
M_2	12.6	10.9	22.3	14.3	60
M_3	12.0	12.0	16.0	12.0	52
M_5	9.4	12.5	12.5	25.0	59

	score classes (%)				
$\geq 10\%$	M_0	M_1	M_2	M_3	M_5
article (9)	11.1	22.2	22.2	22.2	22.2
bdy (9)	22.2	11.1	22.2	22.2	22.2
sec (17)	35.3	17.6	11.8	17.6	17.6
ss1 (11)	45.5	9.1	18.2	9.1	18.2
$\Sigma \geq 60\%$ (9)	44.4	11.1	22.2	11.1	11.1

the topics where our system scores high, there are relatively few elements assessed relevant, at most 18. The other direction of our hypothesis is however not supported by the data. While low assessment count seems to be necessary for high score, it is not sufficient. This can be seen from Table 5 (right). There are quite a few topics that have few assessments (≤ 20), but our system does not score well. Note, however, that we can quite safely say that if there are many assessments then our system will score badly.

We now turn to our second hypothesis: *Our system scores well if, and only if, the assessor likes somewhat large elements.* We look at the 6 most frequent tag-names appearing in the assessments. We will define 4 of them to be "somewhat large," namely article, bdy, sec and ss1. We refer to the set of these tag-names as Σ. The other two frequent tag-names, p and it, we do not regard as "somewhat large." First we try to see if it is true that *if our system scores well then the assessor likes somewhat large elements.* Table 6 (left) shows, for each score class, the percentage of assessments belonging to the most frequent "somewhat large" tag-names. For the topics with the highest precision, M_5, we see that almost 60% of the relevant elements belongs to Σ. For the next two classes the percentage is 52% and 60%, respectively. For the classes of topics where we score lower than 0.1 the percentage is 33% and 15%, respectively. So far the data seems to give some support for our hypothesis. Let's now look at the other direction: *if the assessor likes somewhat large elements then our system scores well.* This direction is not supported by the data, however. We take a look at the "somewhat large" elements and for each element-type we look at the topics for which 10% or more of the relevant elements have the particular tag-name. Table 6 (right) shows, for each tag-name the distribution of those topics over our score-based classes. The bottom line of the table shows the distribution of the topics for which more than 60% of assessments have tag-names from Σ. The topics are relatively evenly distributed over our score-based classes and thus does not support that direction of our hypothesis.

Now it's time for our third hypothesis: *Our system scores well if, and only if, the recall base of the topic is overlapping.* Table 7 shows the relation between

Table 7. Relation between overlapping recall base and retrieval effectiveness. (Left): For each score class, the distribution of topics over overlap classes. (Right): For each overlap class, the distribution of topics over score classes. The count of topics in each overlap class is in brackets.

	overlapping recall-base (%)				
	0-20	21-40	41-60	61-80	81-100
M_0	8.3	16.7	33.3	16.7	25.0
M_1	–	–	25.7	–	75.0
M_2	33.3	–	33.3	33.3	–
M_3	–	–	–	66.7	33.3
M_5	33.3	–	–	66.7	–

	score classes (%)				
	M_0	M_1	M_2	M_3	M_5
0-20 (3)	33.3	–	33.3	–	33.3
21-40 (2)	100	–	–	–	–
41-60 (6)	66.7	16.7	16.7	–	–
61-80 (7)	28.6	–	14.3	28.6	28.6
81-100 (7)	42.9	42.9	–	14.3	–

system scoring and overlapping recall base. The left table shows for each of the score classes, the distribution over overlapping classes. We can see that for two-thirds of our best scoring topics, the overlap is 61–80%. We can thus say that if we score well then there is rather high overlap in the recall base. As with our other hypotheses the other direction is not supported by the data. High overlap in the recall base does not lead to high scoring for our system. What we can say, however, is that if the overlap is rather high (60–80%), we score quite well. For the overlap classes above and below, we score poorly.

What have we learned from the analysis in this section? We have seen that divergence in scores is very great from one topic to another. We have argued that this might be due to lack of assessor agreement. This disagreement is possibly an artifact of a combination of the unclear nature of the XML element retrieval task and the complex assessment procedure. Bluntly, assessors must perform an unclear task with machinery that is too complex to use.

We have also seen some characteristics of the topics where we score well: there are few relevant elements, the elements are fairly long and quite overlapping. While we could say that the first two characteristics are, respectively, understandable and intuitive, we might be tempted to judge the third one as disappointing and contrary to the whole purpose or XML element retrieval. But before we make such judgments we should recall what the task at hand was.

In the analysis above, we have made some crucial assumptions. We defined a particular task by fixing the assessments and evaluation metric. Based on those assumptions we have tried to analyze the performance of our system. For most test collections, this would not raise any methodological questions. The assessment procedure and evaluation metrics are usually not subject to debate. In the INEX case, those issues are far from being settled. The assessments, the metrics, and, in particular, the interplay between assessments and metrics, are all open questions. We think, however, that the kind of analysis that we carried here may yield valuable insights into what is happening under the hood of an XML element retrieval engine. If we can explain why we are unhappy with the reasons why a system scores well, we can more precisely pinpoint the flaws of

the evaluation methodology. In this case we might be unhappy about the fact that for our high scoring topics the overlap in the recall-base is high.

This brings us to the question whether overlap in the recall base is good or bad. The `_eval` measures seem to say it is good, but XCG seems to disagree. It is easy to disagree on this matter. From a user perspective overlap is bad because it puts additional effort on the user who has to look at the same material over and over again (depending, of course, also on the mode of presentation in the interface). From a test collection perspective overlap may be desirable because it is necessary to get a complete pool of relevant elements and we need to be able to reward near misses.

The different perspectives seem to advocate two different retrieval tasks. Currently, it is unclear where the INEX task stands relative to those two. In some sense it seems to be trying to play both roles at the same time. Is this actually possible? One of the urgent priorities for the INEX community is to clearly and unambiguously define what the actual retrieval task is.

6 Conclusions

In this paper, we documented our experiments at the INEX 2004 ad hoc retrieval track. We used a document-element mixture model for processing content-only queries and a more complex mixture approach for handling content-and-structure queries. We investigated the effectiveness of element-based query expansion, and found that it improved retrieval effectiveness. Adding feedback, however, increased overlap. Unfortunately, we do not have access to an implementation of the XCG measure to see how this affects user-oriented evaluation. We investigated the impact of non-overlap on the runs, and found that returning overlapping results leads to far superior scores on all measures, except the recently proposed XCG measure where it was inferior by quite a margin. The radical difference between the measures suggest that they are measuring the quality of two radically different tasks.

We argue that a more explicit definition of the INEX element retrieval task is needed. Our results for the VCAS task showed that, if evaluated in terms of mean average precision, the content-oriented-based approach is clearly superior to a more structured processing of the content-and-structure topics. From the vantage point of a retrieval system, our experiments highlighted the great similarity between the CO and VCAS tasks. If, on the other hand, we evaluate in terms of early precision, the tables turn. Structural processing is superior to looking only at content. This indicates that the great added value of structural constraints is to improve initial precision.

Informed by an analysis of the scoring behavior of one of our runs, we have argued that it is very important for the INEX community to use the data collected to date to clarify the complete INEX process, from the retrieval task, through assessment procedure, to evaluation.

7 Acknowledgments

Jaap Kamps was supported by the Netherlands Organization for Scientific Research (NWO), under project number 612.066.302. Maarten de Rijke was supported by grants from NWO, under project numbers 365-20-005, 612.069.006, 612.000.106, 220-80-001, 612.000.207, 612.066.302, 264-70-050, and 017.001.190.

References

1. T. Grust. Accelerating XPath Location Steps. In *Proc. SIGMOD*, pages 109–120. ACM Press, 2002.
2. D. Hiemstra. *Using Language Models for Information Retrieval*. PhD thesis, University of Twente, 2001.
3. ILPS. The ILPS extension of the Lucene search engine, 2004. http://ilps.science.uva.nl/Resources/.
4. J. Kamps, M. de Rijke, and B. Sigurbjörnsson. Length normalization in XML retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, (SIGIR 2004)*, pages 80–87, 2004.
5. J. Kamps, M. Marx, M. de Rijke, and B. Sigurbjörnsson. The importance of morphological normalization for XML retrieval. In *Proceedings of the First Workshop of the INitiative for the Evaluation of XML retrieval (INEX)*, pages 41–48. ERCIM Publications, 2003.
6. G. Kazai, M. Lalmas, and A. de Vries. The overlap problem in content-oriented XML retrieval evaluation. In *SIGIR '04: Proceedings of the 27th Annual International Conference on Research and Development in Information Retrieval*, pages 72–79. ACM Press, 2004.
7. Lucene. The Lucene search engine, 2004. http://jakarta.apache.org/lucene/.
8. J. Ponte. Language models for relevance feedback. In W. Croft, editor, *Advances in Information Retrieval*, chapter 3, pages 73–96. Kluwer Academic Publishers, Boston, 2000.
9. J. Rocchio, Jr. Relevance feedback in information retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*, chapter 14, pages 313–323. Prentice-Hall, Englewood Cliffs NJ, 1971.
10. B. Sigurbjörnsson, J. Kamps, and M. de Rijke. An Element-Based Approch to XML Retrieval. In *INEX 2003 Workshop Proceedings*, pages 19–26, 2004.
11. B. Sigurbjörnsson, J. Kamps, and M. de Rijke. Processing content-oriented XPath queries. In *Proceedings of the Thirteenth Conference on Information and Knowledge Management (CIKM 2004)*, pages 371–380. ACM Press, 2004.
12. Snowball. The Snowball string processing language, 2004. http://snowball.tartarus.org/.