University of Amsterdam at INEX 2005: Adhoc Track

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

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

Abstract. In this paper we describe the University of Amsterdam's participation in the INEX 2005 adhoc track. Our main research questions for this round of INEX were to investigate selective indexing strategies and different methods for using structural constraints in queries. As a side question, we did experiment with the automatic creation of structured queries.

1 Introduction

In this paper we describe the University of Amsterdam's participation in the INEX 2005 adhoc track. In previous years we have made our runs based on an index of all overlapping XML elements. Our main objective this year was to experiment with different methods of creating a more selective index. The aim is to create a more efficient retrieval system without sacrificing much of retrieval effectiveness. In our experiments with structured queries in the previous years we have found that structural constraints lead to improvements in initial precision. This year we wanted to explore whether different types of structural constraints contribute differently to this gain.

For the *CO.Focussed* task we experiment with different non-overlapping runs. First of all, we retrieve from our full overlapping element index and perform result list based overlap removal as a post processing step. Second, we retrieve from a non-overlapping element index. In our case, we used an index of all `<sec>` elements. Last, and perhaps least, we retrieve from an article index. For the *CO.Thorough* task we experiment with two pruning methods. One based on previous relevance assessments and the other based on the length of the XML elements. We compare the selective indexes to our full overlapping element index. For the *CO.FetchBrowse* task we use our CO.Focussed runs as a basis and simply group results for each article.

For the different *CO+S* tasks we experiment with different extents to use the structural constraints. First of all, we only use the target constraint. Second we use only the field constraints. Third we use both target and field constraints. For the CO queries which do not have a structural version, we introduce structured constraints using pseudo relevance feedback. For efficiency reasons, our system is restricted to handle a limited set of structural constraints. The appropriate set of structural constraints is chosen by studying content-and-structure queries of previous years..

This paper is further organized as follows. In Section 2 we introduce our indexing schemes. We describe our CO and CO+S runs in Sections 3 and 4 respectively. Section 5 gives a summary of our results. Finally we provide some discussion and conclusions in Section 6.

2 Indexing

For effective and efficient XML retrieval indexing plays an important role. Any element can, in theory, be retrieved. It has been shown, however, that not all elements are equally likely to be appreciated as satisfactory answers to an information need [2]. In particular, retrieval of the very many, very small elements is not likely to be rewarded by users. Furthermore, users (and hence metrics) may be willing to partially reward near misses. This prompts us to investigate whether we can reduce our indexing size, both in terms of retrievable units and storage size. We believe that this gives us more efficient retrieval without loosing any, or at least little, of retrieval effectiveness.

Element Indexes For retrieving elements we build four indexes.

- *Element index* We build the "traditional" overlapping element index in the same way as we've done in the previous years (see further [4, 5]).
- *Length based index*: It has been shown that very short elements are not likely to be regarded as relevant. We analyze the average length of elements bearing different tag-names. We then index only element types having an average length above a certain threshold. For INEX 2005 we set the threshold to be 25 terms. The term count was applied before stop-words were removed.
- *Qrel based index*: It has been shown that elements with certain tag-names are more likely than others to be regarded as relevant. We analyze the assessments and look at which elements are assessed more frequently than other. We index only elements that have appeared relatively frequently in previously assessment sets (i.e., they should constitute at least 2% of the total assessments). We index `article`, `bdy`, `sec`, `ss1`, `ss2`, `p`, `ip1`, and `fig`.
- *Section index*: Retrieval of non-overlapping elements is a hot topic in XML retrieval. We want to investigate how simple you can make your non-overlapping retrieval. We build an index based on non-overlapping passages, where the passage boundaries are determined by the structure. The simplest solution is to index only sections (`<sec>`). We believe that this simple strategy is effective, despite (due to) the fact that the sections do not provide a full coverage of the collection.

Article Indexes For retrieving articles we build two indexes.

- *Article index*: the "normal" article index
- *Query fields*: An article index containing both content and a selection of fields. The fields are chosen based on structure of previous structured queries. The fields chosen for INEX 2005 were: `abs`, `fm//au`, `fm//atl`, `kwd`, `st`,

Table 1. Properties of the the different indexes. *Unit* stands for the number of retrievable units. *Storage* stands for the size occupied in physical storage. *Query time* stands for the time needed to retrieve 1000 retrieval units from the index for each of the INEX 2005 topics. All retrieval times are relative to the maximum retrieval time. (This table will be completed in the proceedings version of this paper.)

Index	Units	Storage	Query time
Element index	10,629,617	1.9G	1.0
Length based	1,502,277	1.3G	t.b.a.
Qrel based	1,581,031	1.1G	t.b.a.
Sections	96,600	223M	t.b.a.
Articles	16,819	204M	t.b.a.
Query fields	16,819	275M	t.b.a.

`bb//au`, `bb//atl`, and `ip1`. The fields were chosen from a set of fields that were used in the INEX 2003 and INEX 2004 content-and-structure queries.

For all indexes, stop-words were removed, but no morphological normalization such as stemming was applied. Table 1 shows some statistics of the different indexes.

3 Content-Only Runs

For all our runs we use multinomial language model [1]. We use the same mixture model implementation as we used in INEX 2004 [5]. We assume query terms to be independent, and rank elements according to:

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

where q is a query made out of the terms $t_1, \ldots, t_k$. We estimate the element language model by taking a linear interpolation of three language models:

$$P(t_i|e) = \lambda_e \cdot P_{mle}(t_i|e) + \lambda_d \cdot P_{mle}(t_i|d) + (1 - \lambda_e - \lambda_d) \cdot P_{mle}(t_i), \tag{2}$$

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). Finally, we assign a prior probability to an element e relative to its length in the following manner:

$$P(e) = \frac{|e|}{\sum_e |e|}, \tag{3}$$

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

3.1 CO.Focused

In our focused task we experiment with two different ways of choosing focused elements to retrieve. First, based on the hierarchical segmentation of the collection. Second, based on a linear segmentation of the collection. We also wanted to compare these two approaches with a non-focused baseline, namely a document retrieval system. We submitted three runs:

- *Article run* (UAmsCOFocArticle) A baseline run created using our article index. We used a $\lambda = 0.15$ and a normal length prior.
- *Element run* (UAmsCOFocElements) A run created using a mixture model of the overlapping element index and the article index. We set $\lambda_e = 0.4$ and $\lambda_d = 0.4$. No length prior was used for this run. Overlap was removed in a list-based fashion, i.e. we traversed the list from the most relevant to the least relevant and threw out elements overlapping with an element appearing previously in the list.
- *Section run* (UAmsCOFocSections) A run created using a mixture model of the section index and the article index. We set $\lambda_e = 0.05$ and $\lambda_d = 0.1$. A normal length prior was used.

3.2 CO.Thorough

The main research question is to see if we can get away with indexing only a relatively small number of elements. In our runs we compare three element indexes. The "normal" element index, the qrel-based element selection and the length-based element selection. We submitted three runs:

- *Full element run* (UAmsCOTElementIndex) A run using a mixture model of the full element index and the article index. We set $\lambda_e = 0.05$, $\lambda_d = 0.1$, and used a normal length prior.
- *Qrel-based run* (UAmsCOTQrelbasedIndex) A run using a mixture model of the qrel-based element index and the article index. We set $\lambda_e = 0.05$, $\lambda_d = 0.1$, and used a normal length prior.
- *Length-based run* (UAmsCOTLengthbasedIndex) A run using a mixture model of the length-based element index and the article index. We set $\lambda_e = 0.05$, $\lambda_d = 0.1$, and used a normal length prior.

3.3 CO.FetchBrowse

For the fetch and browse we mirror the focused task submissions, but cluster the results so that elements within the same article appear together.

- *Article run* (UAmsCOFBArticle) This run is exactly the same as the article run we submitted for the focused task.
- *Element run* (UAmsCOFBElements) We took the focused element run and reordered the results in such a way that elements from the same document are clustered together. The document clusters are ordered by the highest scoring element within each document. We returned a maximum of 10 most relevant elements from each article.

– *Section run* (UAmsCOFBSections) We took the focused section run and reordered the result set in such a way that the elements from the same document are clustered together. The document clusters are ordered by the highest scoring section within each document.

4 Content-Only with Structure Runs

For the CO+S task we experiment with three ways of using structural constraints.

Target-only For queries that have a CAS title we only return elements which satisfy the target constraint of the CAS title. For queries that ask for sections, we accept the equivalent tags as listed in the topic development guidelines. NB! We use the terms in the title field of the queries because we want a direct comparison to CO runs. Retrieval is performed using a mixture model using the overlapping element index and the normal document index.

Fields-only Here we use the document index with query fields. We process the queries in three different ways, depending on their format. First, for the the `<castitle>` queries with field constraints that match our fielded article index, we rewrite the query such that it fits our index. For example, the query:

```
//article[about(.//abs, ipv6)]//sec[about(., ipv6 deployment) or
about(., ipv6 support)]
```

becomes

```
abs:ipv6 ipv6 deployment ipv6 support.
```

For the queries that only partly match our indexing scheme, we do additional processing, i.e.

```
//*[about(.//au, moldovan) and about(., semantic networks)]
```

becomes

```
fm//au:moldovan bb//au:moldovan semantic networks
```

since our index makes distinction between article authors and referenced authors. Second, for `<castitle>` queries that do not have fields that fit our index, we use the simply extract the query terms. I.e.

```
//article[about (.//bdy, synthesizers) and about (.//bdy, music)]
```

becomes

```
synthesizers music.
```

Third, for queries that do not have a `<castitle>`, we add structured query fields using pseudo relevance feedback on the fielded article index [3]. We look at the top 20 feedback terms and we add up to n fielded terms where n is the length of the original query. For example,

```
computer assisted composing music notes midi
```

becomes

```
bb//atl:music bb//atl:musical st:music ip1:musical ip1:music
fm//au:university computer assisted composing music notes midi
```

We use those queries to create an article run using the fielded article index. Now we do the following:

- We take an existing run and for each element in that run, we replace it's score with the score of it's article (using the fielded index and fielded queries).
- We do a combSUM of the original element run and the "article score" element run.

Target and Fields Constraints Here we process both the target and fields constraints in the same ways as discussed above.

4.1 Runs

The ways of processing structural constraints discussed above, are applied to each of the structured retrieval tasks.

+S.Focused

- *Strict on target* (UAmsCOpSFocStrictTarget) A run created using a mixture model of the overlapping element index and the article index. We set $\lambda_e = 0.4$ and $\lambda_d = 0.4$. No length prior was used for this run. Target restriction was implemented for queries that had one. Overlap was removed in a list-based fashion
- *Using constraints* (UAmsCOpSFocConstr) We apply the fields-only approach, described above, on the focused CO element run (UAmsCOFocElements).
- *Using constraints and strict on target* (UAmsCOpSFocConstrStrTarg) We apply the fields-only approach on the strict on target run (UAmsCOpSFoc-StrictTarget).

+S.Thorough

- *Strict on target* (UAmsCOpSTStrictTarget) A run created using a mixture model of the overlapping element index and the article index. We set $\lambda_e = 0.05$ and $\lambda_d = 0.1$. We apply a normal length prior. Target constraints are respected for queries that have one.

– *Using constraints* (UAmsCOpSTConstr) We apply the fields-only approach, described above, on the thorough CO element run (UAmsCOTElementIndex).
– *Using constraints and strict on target* (UAmsCOpSTConstrStrTarg) We apply the fields-only approach on the strict on target run (UAmsCOpSTStrictTarget).

+S.FetchBrowse

– *Strict on target* (UAmsCOpSFBStrictTarget) We reorder the focused strict on target run (UAmsCOpSFocStrictTarget) such that results from the same article are clustered together. Only the 10 most relevant elements are considered for each article.
– *Using constraints* (UAmsCOpSFBConstr) We reorder the focused run using constraints (UAmsCOpSFocConstr) such that results from the same article are clustered together. Only the 10 most relevant elements are considered for each article.
– *Using constraints* (UAmsCOpSFBConstrStrTarg) We reorder the focused run using constraints and strict targets (UAmsCOpSFocConstrStrTarg) such that results from the same article are clustered together. Only the 10 most relevant elements are considered for each article.

5 Results

In this section we will present and discuss our preliminary results. The results for the Focussed and Thorough tasks are based on results from the INEX (lip6) website as they appeared on November 6th, 2005. The results for the FetchBrowse were taken from the website in November 10th, 2005.

5.1 The Focussed Task

Table 2 shows the results for both the CO.Focussed and CO+S.Focussed runs.

CO.Focussed The aim of the CO.Focussed submission was to compare 3 non-overlapping retrieval strategies: element retrieval, section retrieval and article retrieval. As we see in Table 2 the element based retrieval outperforms the other two for almost all metrics. There are, however, some notable exceptions. Interestingly, the article run outperforms the other two when we look at extremely early precision of the generalized nxCG metric. For the generalized extended Q and R metric the section run gives the best performance, followed by the article run. This suggests that the element retrieval approach is a good approach when considering the full recall base. However, for early precision, section retrieval (and to some extent article retrieval) is a better alternative.

Table 2. Results for the CO.Focussed and COS.Focussed runs using various metrics

(a) nxCG (overlap=on, generalized)

Run	MAnxCG	@1	@2	@3	@4	@5	@10	@50	@100
Elements	.269	.195	.191	.221	.204	.209	.200	.165	.181
Sections	.204	.213	.209	.190	.194	.178	.176	.158	.153
Articles	.098	.248	.250	.205	.191	.184	.170	.102	.089
StrTarg	.225	.236	.230	.246	.236	.230	.224	.168	.168
Constr	.300	.161	.215	.224	.232	.215	.203	.170	.190
ConStrTar	.237	.289	.272	.257	.250	.231	.224	.182	.175

(b) EP/GR (overlap=on, generalized) and Extended Q and R (generalized)

	EP/GR		Ext. Q and R	
Run	iMAep	MAep	Q	R
Elements	.056	.071	.115	.159
Sections	.030	.064	.145	.215
Articles	.020	.048	.133	.195
StrTarg	.049	.072	.135	.202
Constr	.059	.074	.123	.166
ConStrTar	.057	.078	.144	.208

CO+S.Focussed The aim of the CO+S.Focussed submission was to compare different extents to which the structural constraints can be used. For the averaged metrics the run using only the fields-only approach (Constr) outperforms both the other runs using structured queries, as well as outperforming the runs using no structure at all. For the early precision metrics, the runs using strict-target interpretation (StrTarg and ConStrTarg) outperform the fields-only approach. The run using both field-constraints and target-constraints generally outperforms the run using target-constraints only. It seem thus that the field-constraints are generally useful for improving retrieval effectiveness, while the target constraints are particularly useful for achieving high early precision.

5.2 The Thorough Task

Table 3 shows the results for the CO.Thorough and CO+S.Thorough runs.

CO.Thorough The aim of the CO.Thorough submission was to experiment with two selective indexing approaches compared to a full element retrieval approach. Table 3 shows that for the averaged metrics, the selective indexing approaches to not have substantially worse performance than the full element retrieval approach. Furthermore, for the early precision metrics, the selective indexing approaches outperform the full element retrieval approach.

CO+S.Thorough The aims and the results for the CO+S.Thorough task are the same as for the CO+S.Focussed task. Field-constraints are useful over-all, but target constraints improve initial precision.

Table 3. Results for the CO.Thorough and CO+S.Thorough runs using various metrics

(a) nxCG (overlap=off, generalized)

Run	MAnxCG	@1	@2	@3	@4	@5	@10	@50	@100
Element	.309	.239	.191	.256	.265	.275	.265	.230	.218
Qrel	.301	.266	.227	.293	.287	.289	.275	.245	.231
Length	.302	.281	.247	.281	.272	.276	.264	.240	.218
StrTarg	.192	.322	.260	.263	.245	.246	.225	.186	.168
Constr	.334	.242	.244	.249	.250	.254	.261	.234	.246
ConStrTar	.206	.311	.264	.246	.242	.235	.216	.198	.180

(b) EP/GR (overlap=off, generalized) and Extended Q and R (generalized)

	EP/GR		Extended Q and R	
Run	iMAep	MAep	Q	R
Element	.072	.085	.162	.269
Qrel	.074	.089	.168	.275
Length	.070	.085	.166	.272
StrTarg	.046	.053	.098	.193
Constr	.078	.089	.166	.270
ConStrTar	.049	.056	.101	.193

Table 4. Results for the CO.FetchBrowse and CO+S.FetchBrowse runs using various metrics

(a) ERPRUM (ideal: GK-SOG, quant: Exh, behaviour: Hierarchic)

Run	Average	@1	@5	@10	@20	@100
Elements	.091	.089	.039	.025	.016	.005
Sections	.114	.123	.056	.031	.019	.005
Articles	.027	.072	.033	.021	.012	.003
StrTarg	.107	.098	.039	.026	.016	.005
Constr	.064	.106	.047	.029	.019	.006
ConStrTar	.070	.116	.044	.027	.017	.005

5.3 The Fetch-and-Browse Task

FetchBrowse Table 4 shows the results for our FetchBrowse submissions. At the time of writing, none of the FetchBrowse tasks have been evaluated with an official INEX metrics which handled overlap. Since we submitted only non-overlapping runs for this task, we report the EPRUM metric which takes overlap into consideration. Our section retrieval run outperforms the full element retrieval run both w.r.t. average precision and initial precision. It is interesting to note that for the Focussed task, the element run outperformed the section run for the averaged metric. Note, however, that there are several crucial differences between the results for the two tasks. First of all, the task is of course different. Second, the tasks are evaluated with different metrics. Third, for articles where more than 10 results were found, results 11 and above were removed from the FetchBrowse runs.

Table 5. Results for the CO.FetchBrowse-D and CO+S.FetchBrowse-D runs using various metrics

(a) inex_eval

		generalized				
Run	MAP	@1	@5	@10	@20	@100
Element	.186	.261	.207	.183	.154	.064
Sections	.264	.359	.289	.216	.163	.064
Article	.282	.424	.285	.260	.202	.069
StrTarg	.186	.293	.189	.171	.128	.057
Constr	.242	.315	.239	.216	.175	.068
ConStrTar	.250	.326	.265	.225	.178	.066

FetchBrowse-D Table 5 shows the document-run evaluation of our FetchBrowse submissions. The results seem to indicate that element retrieval is not an effective strategy for improving document retrieval. These results will be analyzed further in the proceedings version of this paper.

6 Conclusions

In INEX 2005 we set out to investigate several research questions.

- Does retrieval using the XML tree hierarchy improve significantly over using a simpler linear segmentation of the documents?
- How do different types of structural constraints contribute to improved retrieval effectiveness?
- Can we prune our overlapping element index to gain efficiency without loosing effectiveness?
- Can we create structured queries automatically using pseudo relevance feedback?

Our results show that retrieving from the full hierarchy of element outperforms retrieval from a linear segmentation. The segmentation based retrieval is however competitive when we look at initial precision. Article retrieval is interestingly effective at P@1 and P@2 for the CO.Focused task.

We showed that fielded constraints are helpful for improving average retrieval performance. Interpreting target constraints in a strict manner does hurt average performance. The target constraints do however improve retrieval when we look at early precision.

For the thorough task we experimented with two different pruning of the full overlapping element index. Neither of the pruning strategies lead to a considerably lower average performance. Both pruning strategies did however lead to improved initial precision. The length based pruning lead to greater improvement than the qrel based pruning.

For the FetchBrowse task the linear segmentation performed considerably better than the hierarchical segmentation. This result is different from the Fo-

cussed task. It is however not clear whether this difference lies in different task performed or in the different metric used to evaluate the two tasks.

For the document ranking part of the FetchBrowse task, ranking documents based on their own retrieval score outperformed the document retrieval based on the highest scoring element/section.

At this point we have not evaluated the effect of automatically creating structured queries through pseudo relevance feedback. This analysis remains as future work and will be carried out before publication of the proceedings version of this paper.

Future work includes looking at different granularities for the linear segmentation. Instead of looking at section level, we could look at using subsections, when available. Also we should include elements such as front-matter so that the linear segmentation has a full coverage of the collection.

Our processing of the CO+S queries is a bit ad-hoc. Future research should include a cleaner way of processing and retrieving using structural queries.

Acknowledgments

This research was supported by the Netherlands Organization for Scientific Research (NWO) under project numbers 017.001.190, 220-80-001, 264-70-050, 612-13-001, 612.000.106, 612.000.207, 612.066.302, 612.069.006, and 640.001.501.

References

1. D. Hiemstra. *Using Language Models for Information Retrieval*. PhD thesis, University of Twente, 2001.
2. J. Kamps, M. de Rijke, and B. Sigurbjörnsson. The importance of length normalization for XML retrieval. *Information Retrieval*, 8:631–654, 2005.
3. 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.
4. B. Sigurbjörnsson, J. Kamps, and M. de Rijke. An Element-Based Approch to XML Retrieval. In *INEX 2003 Workshop Proceedings*, pages 19–26, 2004.
5. B. Sigurbjörnsson, J. Kamps, and M. de Rijke. Mixture models, overlap, and structural hints in XML element retreival. In *Advances in XML Information Retrieval*, LNCS 3493, pages 196–210, 2005.