The University of Amsterdam at INEX 2006

Jaap Kamps[1,2], Marijn Koolen[1], and Börkur Sigurbjörnsson[2]

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

Abstract. We describe the University of Amsterdam's participation in the INEX 2006 Adhoc Track. We participated in all four Adhoc Track tasks, and report initial findings based on a single set of measure for all four tasks. Our main findings are the following. First, a complete element index outperforms a restricted index based on section-structure, albeit the differences are small. Second, grouping elements per article does not lead to performance degradation, but may improve scores. Third, all restrictions of the "pure" element runs (by removing overlap, by grouping elements per article, or by selecting a single element per article) lead to some but only moderate loss of precision.

1 Introduction

In this paper we document the University of Amsterdam's participation in the INEX 2006 Adhoc Track. Our main aims for INEX 2006 were to investigate the effectiveness of our XML retrieval approaches on a new collection, the Wikipedia XML corpus [1], which has a different nature than the IEEE collection used in INEX 2002–2005. What are the characteristics of the new Wikipedia collection, and how do they affect the performance on our element retrieval system? We want to know which approaches tranfer well to a new sort of collection, and which approaches don't and why.

The rest of the paper is organized as follows. First, Section 2 describes the Wikipedia collection. Next, Section 3 documents the XML retrieval system used in the experiment. Then, in Section 4, we detail the setup of the experiments. The results of the experiments are reported in Section 5. Finally, in Section 6, we discuss our findings and draw some initial conclusions.

2 Wikipedia collection

In previous years, the IEEE collection was used in INEX. This year sees the introduction of a new collection, based on the English Wikipedia collection [2]. The collection has been converted from the wiki-syntax to an XML format [1]. Whereas the IEEE collection has somewhat over 12,000 documents, the Wikipedia collection has more than 650,000 documents. To get some idea of the characteristics of this new collection, we have gathered some statistics. Table 1 shows a few basic collection statistics. There are over 50,000,000 elements using 1241

different tag names. However, of these, 779 tags occur only once, and only 120 of them occur more than 10 times in the entire collection. On average, documents have almost 80 elements, with an average depth of 4.82.

Table 1. Wikipedia collection statistics

Description	Statistics
# of articles	659,388
# of elements	52,555,826
# of unique tags	1241
Avg. # of elements per article	79.69
Average depth	4.82

Next, we gathered tag statistics like collection frequency, document frequency and element length. Table 2 shows the 10 longest elements and their collection frequency. The length is the average number of words in the element. The `<article>` element is the longest element of course, since it always encompasses all other elements. However, after the `<body>` element, the other long elements occur only rarely in the entire collection, and contain only a few hundred words. Clearly, most of the elements are rather short. Even the average article length is short, containing no more than 415 words.

Table 2. Longest elements in Wikipedia collection

Element	Mean length	Collection freq.
`<article>`	414.79	659,388
`<body>`	411.20	659,388
`<noinclude>`	380.83	14
`<h5>`	253.18	72
`<td_align>`	249.20	4
`<h4>`	237.13	307
`<ol>`	198.20	163
`<timeline>`	186.49	48
`<number>`	168.72	27
`<h3>`	163.80	231

In table 3 the most frequent tag names are listed. Column 2 shows the average document frequency of the tag name, column 3 shows the collection frequency. There are many links to other Wiki pages (`<collectionlink>`s), and many `<unknownlink>`s that are not really links (yet). Wiki pages have more than 4 paragraphs (indicated by `<p>` tags) and more than 2 sections on average.

As shown in table 4, the elements `<article>`, `<conversionwarning>`, `<body>` and `<name>` occur in every single document. Almost all documents have links to

Table 3. Most frequent tags in Wikipedia collection

Tag name	Document freq.	Collection freq.
<collectionlink>	25.80	17,014,573
<item>	8.61	5,682,358
<unknownlink>	5.98	3,947,513
<cell>	5.71	3,770,196
<p>	4.17	2,752,171
<emph2>	4.12	2,721,840
<template>	3.68	2,427,099
<section>	2.44	1,609,725
<title>	2.41	1,592,215
<emph3>	2.24	1,480,877

other Wiki pages (99.4%), and more than 70% have text tagged as <unknownlink> (indicating a topic that could have its own page). Together with the average frequency of the <collectionlink>s, this indicates a very dense link structure. Apart from that, the textual unit indicating elements <section> and <p> (paragraph) occur in 69.6% and 82.1% of the documents respectively.

Table 4. Elements with the highest document frequency in Wikipedia collection

Tag name	Document freq.	%
<article>	659,388	100.0
<conversionwarning>	659,388	100.0
<body>	659,388	100.0
<name>	659,388	100.0
<collectionlink>	655,561	99.4
<emph3>	587,999	89.2
<p>	541,389	82.1
<unknownlink>	479,830	72.8
<title>	459,253	69.6
<section>	459,252	69.6

The main observation is that elements are small on average. One important reason for this is the Wikipedia policy of splitting long articles into multiple new pages.[3] The idea is that encyclopedia entries should be focused. If the article grows too long, it should be split into articles discussing the sub-topics. This is

[3] As http://en.wikipedia.org/wiki/Wikipedia:Summary_style reads: "The length of a given Wikipedia entry tends to grow as people add information to it. This cannot go on forever: very long entries would cause problems. So we must move information out of entries periodically. This information should not be removed from Wikipedia: that would defeat the purpose of the contributions. So we must create new entries to hold the excised information." (November 2006).

a policy that closely resembles the main purpose of element retrieval: a relevant results must be specific. The dense structure of the collection links should make it easy to navigate to other relevant pages.

3 XML Retrieval System

3.1 Indexing

Our indexing approach is based on our earlier work [3,4,5]).

- *Element index*: Our main index contains all retrievable elements, where we index all textual content of the element including the textual content of their descendants. This results in the "traditional" overlapping element index in the same way as we have done in the previous years [3].
- *Section index*: We built an index based on frequently retrieved elements. Studying the distribution of retrieved elements, we found that the `<article>`, `<body>`, `<section>` and `<p>` elements are retrieved far more often than other elements. The only exceptions are the `<collectionlink>` elements. However, since collection links contain only a few terms at most, and say more about the relevance of another page, we didn't add them to the index.
- *Article index*: We also build an index containing all full-text articles (i.e., all wikipages) as is standard in IR.

For all indexes, stop-words were removed, but no morphological normalization such as stemming was applied. Queries are processed similar to the documents, we use either the CO query or the CAS query, and remove query operators (if present) from the CO query and the about-functions in the CAS query.

3.2 Retrieval

For all our runs we used a multinomial language model [6]. We use the same mixture model implementation as we used in earlier years [4]. 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. For a more thorough description of our retrieval approach we refer to [4].

4 Experimental Setup

In this section, we detail the experiments and runs submitted to the INEX 2006 Adhoc track tasks. None of our official submissions used the three layered mixture model, i.e., we use $\lambda_d = 0$ throughout.

4.1 Thorough

For the Thorough Task, we submitted two runs using the CO query (from the topic's `<title>` field) and two runs using the CAS query (from the topic's `<castitle>` field). We regard the Thorough Task as underlying all other tasks, and all other runs are based on postprocessing them in various ways.

The two Thorough CO runs are:

`thorough_element_lm` Language model ($\lambda_e = 0.15$) on the element index.
`thorough_section_lm` Language model ($\lambda_e = 0.15$) on the section index.

Our two CAS query runs are also based on postprocessing the CO run based on the element index. We extract all path-restrictions on the element of request in the CAS query, and filter the results for elements conforming on all or some of the location steps.

The two Thorough CAS query runs are:

`thorough_element_lm_cas.joined` Language model ($\lambda_e = 0.15$) on the element index, retaining elements that satisfy the complete path expression.
`thorough_element_lm_cas.seperate` Language model ($\lambda_e = 0.15$) on the element index, retaining element that satisfy at least the tagname of the element of request.

4.2 Focused

For the Focused Task we submitted two runs using the CO query and two runs using the CAS query. All our Focused Task submissions correspond to a Thorough Task submission, and are postprocessed by a straightforward list-based removal strategy. We traverse the list top-down, and simply remove any element that is an ancestor or descendant of an element seen earlier in the list. For example, if the first result from an article is the article itself, we will not include any further element from this article.

The resulting two Focused CO runs are:

`focused_element_lm` Language model ($\lambda_e = 0.15$) on the element index, with list-based removal of ancestor or descendant elements.

`focused_section_lm` Language model ($\lambda_e = 0.15$) on the section index, with list-based removal of ancestor or descendant elements.

The resulting two Focused CAS runs are:

`focused_element_lm_cas.joined` Language model ($\lambda_e = 0.15$) on the element index, retaining elements that satisfy the complete path expression, and with list-based removal of ancestor or descendant elements.

`focused_element_lm_cas.seperate` Language model ($\lambda_e = 0.15$) on the element index, retaining element that satisfy at least the tagname of the element of request, and with list-based removal of ancestor or descendant elements.

4.3 All in Context

For the All in Context Task, we only submitted runs using CO query. Here, we base our runs on the Thorough Task runs using the section index. We cluster all elements belonging to the same article together, and order the article clusters either by the highest scoring element, or by the combined scores of all elements belonging to the article.

The two All in Context CO runs are:

`all_section_lm.highest` Language model ($\lambda_e = 0.15$) on the section index, clustered by article and ranked according to the highest scoring element in an article, and with list-based removal of ancestor or descendant elements.

`all_section_lm.sum` Language model ($\lambda_e = 0.15$) on the section index, clustered by article and ranked according to the sum of element scores in an article, with list-based removal of ancestor or descendant elements.

4.4 Best in Context

Finally, for the Best in Context Task we submitted three runs, all based on the CO query. We use, again, the runs made against the section index, and postprocess them such that only a single result per article is kept in the result file.

`best_section_lm.highest_score` Language model ($\lambda_e = 0.15$) on the section index, selecting only the highest scoring element per article.

`best_section_lm.article` Language model ($\lambda_e = 0.15$) on the section index, selecting the article node of each element from an unseen article.

`best_section_lm.first` Language model ($\lambda_e = 0.15$) on the section index, selecting only the first element (in reading order) that is retrieved per article.

5 Results

At the time of writing, only partial results are available. We opt to compare all runs on equal grounds, focusing on two common measures that are available in the EvalJ package: a mean-average-precision measure (MAep), and early precision measure (nxCG at rank 5, 10, 25, and 50). This allows us to measure the effectiveness of various post-processing methods, such as overlap-removal or clustering by article, in terms of their relative impact on precision and recall. Before discussing the results for each of the Adhoc tasks, we first give some detail about the topics and resulting relevance judgments.

5.1 Topics and Judgments

Assessments are available for 111 topics (numbered 289–298, 300–306, 308–369, 371–376, 378–388, 390–392, 395, 399–407, 409–411, and 413). There is a total 8,737 relevant passages for these 111 topics in 5,483 different articles. Table 5 shows some statistics of the relevant passages (i.e., the text highlighted as relevant by the assessors). It is interesting to see that most relevant passages are

Table 5. Relevant passage statistics

Description	Statistics
# articles with relevance	5,483
# relevant passages	8,737
avg. rel. pass. length	1,098
median rel. pass. length	56

very short. The lengths of the elements are measured in characters (text offset). The length distribution is skewed: the difference between mean and median relevant passage length is enormous. Apparently, most relevant passages contain only a few words or a sentence, indicating that even though the average article is rather short, there is still a lot of irrelevant text that can be filtered out.

Table 6 looks at the judgments from the vista point of elements containing only relevant text. We see that there are many `<collectionlink>` elements that contain relevant text. This is not very surprising, because the `<collectionlink>` element is by far the most frequent element in the entire collection (see Table 3). Other short elements, like `<cell>`, `<emph2>` and `<unknownlink>` are also found often in relevant passages. The lengths mentioned are the average lengths of the *relevant* elements of that type. Longer elements containing relevant text are mostly `<section>` and `<p>` elements.

The shorter elements often contain only a few words, and often are only a small part of the entire passage. However, there are a lot of `<collectionlink>` elements that encompass an *entire* relevant passage. Table 7 shows the frequency

Table 6. Relevant element statistics

Tag name	Frequency	Avg. length
`<collectionlink>`	171,766	14
`<item>`	35,107	285
`<cell>`	29,711	17
`<p>`	29,199	470
`<emph2>`	28,260	22
`<unknownlink>`	24,893	17
`<section>`	20,667	2,434
`<emph3>`	11,867	16
`<row>`	10,148	57
`<title>`	9,082	25

Table 7. Elements encompassing entire relevant passages

Tag name	Frequency	Avg. length
`<p>`	2,813	509
`<collectionlink>`	1,592	16
`<name>`	886	21
`<title>`	715	21
`<emph3>`	699	19
`<item>`	532	140
`<emph2>`	216	60
`<body>`	209	5,227
`<unknownlink>`	202	18
`<caption>`	191	72

of elements that are the shortest element to encompass an entire relevant passage. Apparently, topic authors often consider a link to another page to be a relevant passage. However, the <p> element now surfaces as the most frequent shortest element to encompass an entire relevant passage. This gives support to our Section index as a viable indexing strategy. The focus of this year's relevance metrics is in specificity, though, so these results might point us in the wrong direction.

5.2 Thorough

We'll now discuss the results for the four Adhoc tasks, starting with the Thorough Task. The Thorough Task puts no restriction on XML elements to return. Table 8 shows the results for the Thorough Task. We first discuss the top two

Table 8. Results for the Thorough Task (generalized, off)

Run	nxCG@5	nxCG@10	nxCG@25	nxCG@50	MAep
thorough_element_lm	0.4120	0.3789	0.3262	0.2790	0.0343
thorough_section_lm	0.3948	0.3721	0.2977	0.2503	0.0227
thorough_element_lm_cas.joined	0.1872	0.1642	0.1410	0.1100	0.0116
thorough_element_lm_cas.seperate	0.2124	0.1761	0.1511	0.1208	0.0129

runs using the keyword or CO query. We make a few observations: First, we see that the index containing all XML elements in the collection is more effective on all measures. Second, we see that difference with the section index (containing only article, body, section, and paragraph nodes) is relatively small, especially in terms of precision. This is in line with earlier results on the IEEE collection, and shows the potential of the much smaller section index. We now zoom in on the bottom two runs using the structured or CAS query. We see that the joined run (using the element of request's full path as a Boolean filter) performs less than the separate run (filtering only for the tagname of the element of request). When comparing the results for the CO and CAS queries, we see that the CO query runs are more effective for both mean average precision, and for early precision. While the loss of mean average precision can be expected, the structural hints hold the potential to improve precision. We should note, however, that the CAS processing was done naively, resulting in many topics with very few or no results left.

5.3 Focused

For the Focused Task, none of the retrieved elements was allowed to contain text that overlaps with another retrieved element. We evaluate runs here using the same measures as the Thorough Task above, but since the elements judged relevant in recall base may overlap, performance can never obtain perfect scores.

Table 9. Results for the Focused Task (generalized, off)

Run	nxCG@5	nxCG@10	nxCG@25	nxCG@50	MAep
`focused_element_lm`	0.3571	0.3245	0.2560	0.2116	0.0105
`focused_section_lm`	0.3386	0.2868	0.2212	0.1825	0.0080
`focused_element_lm_cas.joined`	0.1522	0.1310	0.0975	0.0740	0.0033
`focused_element_lm_cas.seperate`	0.1781	0.1418	0.1060	0.0789	0.0037

Table 9 shows the results for the Focused Task. Since we use the same post-processing method—the list-based, top-down removal of elements overlapping with earlier seen text—we see the same relative behavior as for the Thorough Task runs above. The complete element index is still more effective than the smaller section index. This signals that the effectiveness of the element index is not due to the fact that it contains all potentially overlapping elements (which could be exploited in theory). When comparing the Focused Task results to the Thorough Task results, we note that, as expected, the scores are lower. There is a moderate decline for the precision scores, but the recall (and mean average precision) drops dramatically.

5.4 All in Context

For the All in Context Task, there is the further restriction that retrieved elements must be grouped per article (and still may not overlap). Again, we evaluate runs here using the same measures as the Thorough Task above, so optimal performance will result in still imperfect scores. Table 10 shows the results for the

Table 10. Results for the All in Context Task (generalized, off)

Run	nxCG@5	nxCG@10	nxCG@25	nxCG@50	MAep
`all_section_lm.highest`	0.3357	0.3082	0.2290	0.1889	0.0082
`all_section_lm.sum`	0.2454	0.2135	0.1804	0.1470	0.0059

All in Context Task. We see that for ranking the groups of elements from the same article, the best scoring element is a more useful criterion than the sum of all element scores. When comparing the All in Context Task results to the Focused Task results, we note that the precision scores are in the same league. In fact, considering that our All in Context runs are all based on the section index, the clustering by article improves performance for all measures except for the nxCG at rank 5.

5.5 Best in Context

For the Best in Context Task, we may only retrieve a single result per article. For this task, a best-entry-point was obtained from the human judge during the assessment procedure. At the time of writing, the measure corresponding to this best-entry-point judgment is unavailable, and hence we evaluate the retrieved element in terms of perceived topical relevance. For ease of comparison, we evaluate runs here using the same measures as the Thorough Task above, so optimal performance will result in still grossly imperfect scores. Table 11 shows

Table 11. Results for the Best in Context Task (generalized, off)

Run	nxCG@5	nxCG@10	nxCG@25	nxCG@50	MAep
`best_section_lm.highest_score`	0.3290	0.2796	0.2082	0.1670	0.0069
`best_section_lm.article`	0.2451	0.1983	0.1423	0.1106	0.0045
`best_section_lm.first`	0.3290	0.2796	0.2082	0.1670	0.0069

the results for the Best in Context Task. It is comforting to note that the element selection strategies outperform the strategy that simply backs off to the whole article. Our Best in Context runs where based on postprocessing the All in Context run, result in the same performance for selecting the first or highest scoring element. This may be due to the layered processing of the runs, where the Thorough Task's section index run is processed by removing overlap, then clustered by article, and then, finally, we selecting our final element to retrieve. When comparing the Best in Context Task results to the All in Context Task results, we note that there is only a moderate loss of precision for the Best in Context Task.

The main aim of the Best in Context task is to investigate how the chosen best entry point related to the perceived relevance in the article. There is obvious value in comparing the results above to the results based on the selected best entry point.

6 Discussion and Conclusions

This paper documents the University of Amsterdam's participation in the INEX 2006 Adhoc Track. We participated in all four Adhoc Track tasks. At the time of writing, only partial result are available. Hence we decide on a single set of measures for all four Tasks, focusing on both precision and mean average precision. This will allow for straightforward comparison between the tasks, but may not reflect best each individual Task. In particular, we evaluate the Best in Context Task not in terms of the best entry point as provided in the assessments, but in terms of the perceived relevance.

Our main findings so far are the following. First, for the Thorough Task, we see that a complete element index was more effective than a restricted index

based on the sectioning structure, although the difference is not large. We also see that the keyword or CO query was more effective than the structured or CAS query. Second, for the Focused Task, we observe a very similar pattern as for the Thorough Task. This is a reassuring result, because it signals that the superior performance of the element index is not due to the fact that it contains many overlapping elements. Third, for the All in Context Task, we find that the clustering per article is in fact improving the performance when compared to the corresponding overlap-free Focused Task runs. Fourth, for the Best in Context Task, we see that element selection outperforms backing off to the whole article, and obtain—perhaps suprisingly—still agreeable precision scores in terms of perceived relevance.

Acknowledgments

This research was supported by the Netherlands Organization for Scientific Research (NWO, grants # 612.066.302, 612.066.513, 639.072.601, and 640.001.501), and by the E.U.'s 6th FP for RTD (project MultiMATCH contract IST-033104).

References

1. Denoyer, L., Gallinari, P.: The Wikipedia XML Corpus. SIGIR Forum **40** (2006) 64–69
2. Wikipedia: The free encyclopedia (2006) http://en.wikipedia.org/.
3. Sigurbjörnsson, B., Kamps, J., de Rijke, M.: An Element-Based Approch to XML Retrieval. In: INEX 2003 Workshop Proceedings. (2004) 19–26
4. Sigurbjörnsson, B., Kamps, J., de Rijke, M.: Mixture models, overlap, and structural hints in XML element retreival. In: Advances in XML Information Retrieval: INEX 2004. Volume 3493 of LNCS 3493. (2005) 196–210
5. Sigurbjörnsson, B., Kamps, J.: The effect of structured queries and selective indexing on XML retrieval. In: Advances in XML Information Retrieval and Evaluation: INEX 2005. Volume 3977 of LNCS. (2006) 104–118
6. Hiemstra, D.: Using Language Models for Information Retrieval. PhD thesis, University of Twente (2001)