Filtering and Clustering XML Retrieval Results

Jaap Kamps[1,2], Marijn Koolen[1], and Börkur Sigurbjörnsson[2,3]

[1] Archives and Information Science, Faculty of Humanities, University of Amsterdam
[2] ISLA, Faculty of Science, University of Amsterdam
[3] Yahoo! Research, Barcelona

Abstract. As part of the INEX 2006 Adhoc Track, we conducted a range of experiments with filtering and clustering XML element retrieval results. Our basic retrieval engine retrieves arbitrary elements from the collection (corresponding to the Thorough Task). These runs are filtered to remove textual overlap between elements (corresponding to the Focused Task). The resulting runs can be clustered per article (corresponding to the All in Context Task). Finally, we select the "best" element for each article (corresponding to the Best in Context Task). 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 non-overlapping 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 overall motivation for INEX 2006 was 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 transfer well to a new sort of collection, and which approaches don't and why.

During INEX 2006, we conducted a range of experiments with filtering and clustering XML element retrieval results. Our basic retrieval engine retrieves arbitrary elements from the collection (corresponding to the Thorough Task). These runs are filtered to remove textual overlap between elements (corresponding to the Focused Task). The resulting runs can be clustered per article (corresponding to the All in Context Task). Finally, we select the "best" element for each article (corresponding to the Best in Context Task).

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 topics and assessments of the INEX 2006 Adhoc Track. In four separate sections, we report our experiments

Table 1. Wikipedia collection statistics

Description	Statistics
# of articles	659,388
# of elements	52,555,826
# of unique tags	1,241
Avg. # of elements per article	79.69
Average depth	4.82

for the Thorough Task (§5); the Focused Task (§6); the All in Context Task (§7); and the Best in Context Task (§8). Experiments with a mixture language model are discussed in Section 9. Finally, in Section 10, we discuss our findings and draw some 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 1,241 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.

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.

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 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 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
	198.20	163
<timeline>	186.49	48
<number>	168.72	27
<h3>	163.80	231

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

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.[4] The idea is that encyclopedia entries should be focused. If the article

[4] 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

grows too long, it should be split into articles discussing the sub-topics. This is 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 containing only the most 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, and we build an index containing just these elements. The frequent element left out is `<collectionlink>`. 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 e 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:

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

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).

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	289

Table 6. Relevant element statistics

Tag name	Frequency	Avg. length
`<p>`	15,315	450
`<item>`	14,375	91
`<emph2>`	14,322	20
`<cell>`	13,898	19
`<section>`	9,291	2,375
`<emph3>`	5,782	16
`<article>`	5,481	9342
`<body>`	5,479	9322
`<title>`	5,197	21
`<template>`	4,107	87

where $P_{mle}(\cdot|e)$ is a language model for element e in document d; $P_{mle}(\cdot|d)$ 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). None of our official submissions used the three layered mixture model proper, i.e., we use $\lambda_d = 0$ unless indicated otherwise. The default value of λ_e is 0.15.

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

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

where $|e|$ is the size of an element e. The β parameter introduces a length bias which is proportional to the element length with $\beta = 1$ (the default setting). For a more thorough description of our retrieval approach we refer to [4]. For comprehensive experiments on the earlier INEX data, see [7].

4 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 short. The lengths of the elements are measured in characters (text offset).

Table 6 looks at the judgments from the vista point of elements containing only relevant text. After the workshop it was decided that the links in the collection are too small to be relevant. This affects the elements `<collectionlink>`,

Table 7. Elements containing entire relevant passages

Tag name	Frequency
<p>	2,585
<body>	1,639
<section>	1,326
<item>	937
<article>	724
<normallist>	301
<name>	267
<collectionlink>	208
<row>	180
<caption>	174

<outsidelink>, <redirectlink>, <unknownlink>, <weblink>, <wikipedialink>. This has quite some effect on the set of relevant elements, for example, in the original qrels there were no less than 78,792 <collectionlink> elements highlighted by the assessors. Although the links are no longer considered exhaustive, there are still quite a number of small elements left, like <emph2>, <cell> and <emph3> (see Table 3). The lengths mentioned are the average lengths of the *relevant* elements of that type. Longer elements containing relevant text are mostly <p>, <item> and <section> elements.

The shorter elements often contain only a few words, and often are only a small part of the entire passage. However, there are still a fair number of elements that encompass an *entire* relevant passage. Table 7 shows the frequency of elements that are the shortest element to contain an entire relevant passage. There are 208 passages that are fully contained within a <collectionlink> element, and 43 more passages contained by other link elements that are considered too small to be relevant. The passages are not considered too small, so retrieving a larger element containing the passage still gives some score. Relevance is often found at the paragraph level. 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 Experiments for the Thorough Task

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

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

Run	MAep	nxCG@5	nxCG@10	nxCG@25	nxCG@50
`thorough_element_lm`	0.0471	0.4120	0.3789	0.3262	0.2790
`thorough_section_lm`	0.0431	0.3948	0.3721	0.2977	0.2503
`thorough_element_lm_cas.seperate`	0.0265	0.2124	0.1761	0.1511	0.1208
`thorough_element_lm_cas.joined`	0.0222	0.1872	0.1642	0.1410	0.1100

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.

For all submitted runs we used the default length prior parameter $\beta = 1$, although XML retrieval may require special length normalization [8]. Since we're dealing with a new collection, it is interesting to see whether the different parameter settings of the length prior show the same effect on this new collection. We also experimented with different λ_e and β setting to see what works, and what doesn't.

5.1 Results

We will 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 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 good 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.

Table 9 shows the results for non-official runs for the Thorough Task, using different values for the language model parameters. With a higher value of λ_e

Table 9. Results for the Thorough Task (generalized, off) for different parameter settings

Run	MAep	nxCG@5	nxCG@10	nxCG@25	nxCG@50
element $\beta = 0.8, \lambda_e = 0.15$	0.0460	0.3739	0.3383	0.2901	0.2500
element $\beta = 0.9, \lambda_e = 0.15$	0.0465	0.3820	0.3473	0.2940	0.2502
element $\beta = 1.0, \lambda_e = 0.15$	0.0471	0.3868	0.3519	0.2982	0.2508
element $\beta = 1.1, \lambda_e = 0.15$	0.0462	0.3727	0.3317	0.2973	0.2464
element $\beta = 0.8, \lambda_e = 0.30$	0.0462	0.3884	0.3479	0.2966	0.2525
element $\beta = 0.9, \lambda_e = 0.30$	0.0465	0.3897	0.3494	0.3004	0.2533
element $\beta = 1.0, \lambda_e = 0.30$	0.0469	0.3878	0.3512	0.3005	0.2518
element $\beta = 1.1, \lambda_e = 0.30$	0.0465	0.3867	0.3471	0.2995	0.2501

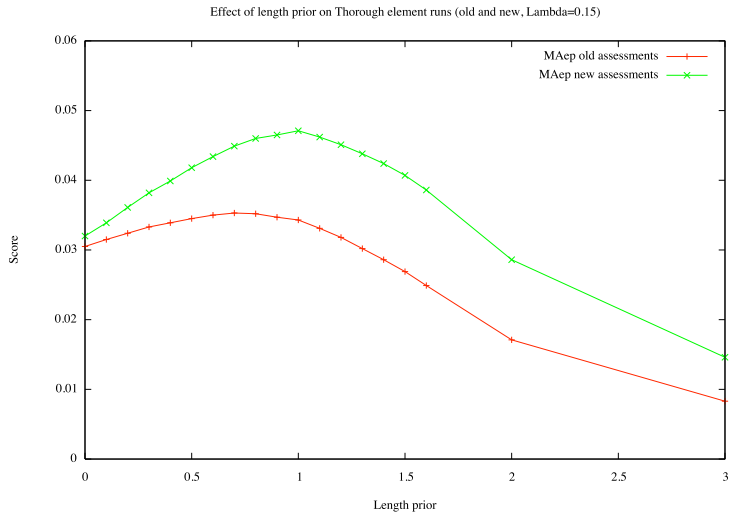

Fig. 1. Effect of the length prior for the Thorough Task using the element index with standard smoothing

we see the expected gain of precision and loss of recall (or MAep). Also, with a higher λ_e value, the length prior parameter β has less effect. The scores for the $\lambda_e = 0.30$ runs are very similar. With the standard $\lambda_e = 0.15$ the standard $\beta = 1.0$ gives the best results.

Figure 1 show the effect of different length prior settings on performance in terms of MAep. We clearly see that the default value $\beta = 1.0$ gives optimal performance with the new assessments. If we compare the new assessments with the old assessments (where the links are considered exhaustive), we see that a lower value of β gives better performance. This can be explained by the huge number of relevant `<collectionlink>` elements in the old assessments. Lower β values retrieve more smaller elements, many of which are `<collectionlink>` elements.

Table 10. Results for the Focused Task (generalized, off)

Run	nxCG@5	nxCG@10	nxCG@25	nxCG@50	MAep
`focused_element_lm`	0.3337	0.2948	0.2252	0.1781	0.0140
`focused_section_lm`	0.3386	0.2868	0.2212	0.1834	0.0152
`focused_element_lm_cas.seperate`	0.2845	0.2288	0.1634	0.1159	0.0080
`focused_element_lm_cas.joined`	0.2400	0.1981	0.1409	0.1010	0.0069

6 Experiments for the Focused Task

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 post-processed 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.

We also look at the impact of different levels of smoothing and of length normalization.

6.1 Results

For the Focused Task, none of the retrieved elements was allowed to contain text that overlaps with another retrieved element. Table 10 shows the results for the Focused Task. The results for the official measure (nxCG) are listed first (columns 2 through 5). The runs based on the section and element index show almost the same performance for this task. At ranks 5 and 50, the section index run performs better than the element index run. For the two runs using the structured query, we see again that the run using only the tagname of the target element ("separate") scores better. The CAS query runs are less effective than the CO query runs, although the difference in performance is much smaller than for the Thorough Task before.

Table 11. Results for the Focused Task (generalized, off) for different parameter settings

Run	nxCG@5	nxCG@10	nxCG@25	nxCG@50	MAep
element $\beta = 0.8, \lambda_e = 0.15$	0.3315	0.2923	0.2305	0.1843	0.0144
element $\beta = 1.0, \lambda_e = 0.15$	0.3337	0.2948	0.2252	0.1781	0.0140
element $\beta = 1.1, \lambda_e = 0.15$	0.3256	0.2913	0.2208	0.1737	0.0135
element $\beta = 0.5, \lambda_e = 0.30$	0.3353	0.3025	0.2354	0.1885	0.0144
element $\beta = 1.0, \lambda_e = 0.30$	0.3410	0.2920	0.2296	0.1790	0.0139
element $\beta = 1.1, \lambda_e = 0.30$	0.3438	0.2913	0.2264	0.1763	0.0139

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. The Thorough measure MAep is listed in column 6 of Table 10 for comparison. Interestingly, the section index run is outperforming the element index run for the Focused Task. When comparing the Focused Task results to the Thorough Task results, we note that, as expected, the scores are substantially lower. There is a moderate decline for the precision scores, but the recall (and mean average precision) drops dramatically.

Table 11 shows the results for non-official runs for the Focused Task, using different values for the language model parameters. As may be expected, a higher value of λ_e leads to a minor increase in precision. However, a lower value of β leads to a small increase at higher ranks and in MAep. This is unexpected since the corresponding Thorough run is actually inferior to the standard length normalization ($\beta = 1.0$). A possible explanation is the non-overlapping nature of the Focused runs: in case a long element is selected, this immediate outlaws a wide range of other elements. A case in point is when the `<article>` element is selected, which effectively exhausts the whole article.

7 Experiments for the All in Context Task

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.

7.1 Results

For the All in Context Task, there is the further restriction that retrieved elements must be grouped per article (and still may not overlap). Table 12 shows

Table 12. Results for the All in Context Task (generalized precision)

Run	MAgP	gP@5	gP@10	gP@25	gP@50
`all_section_lm.highest`	0.1633	0.3014	0.2633	0.1966	0.1579
`all_section_lm.sum`	0.1062	0.1592	0.1351	0.1283	0.1136

Table 13. Results for the All in Context Task (generalized, off)

Run	MAep	nxCG@5	nxCG@10	nxCG@25	nxCG@50
`all_section_lm.highest`	0.0157	0.3357	0.3082	0.2291	0.1898
`all_section_lm.sum`	0.0112	0.2454	0.2135	0.1805	0.1477

Table 14. Results for the All In Context Task (generalized, off) for different parameter settings

Run	MAgP	gP@5	gP@10	gP@25	gP@50
element $\beta = 1.0, \lambda_e = 0.15$	0.1509	0.2743	0.2450	0.1858	0.1449
element $\beta = 1.2, \lambda_e = 0.15$	0.1552	0.2895	0.2611	0.1940	0.1468
element $\beta = 1.4, \lambda_e = 0.15$	0.1556	0.3031	0.2628	0.1947	0.1472
element $\beta = 1.5, \lambda_e = 0.15$	0.1532	0.3046	0.2566	0.1934	0.1452
element $\beta = 1.0, \lambda_e = 0.30$	0.1535	0.2881	0.2489	0.1920	0.1486
element $\beta = 1.2, \lambda_e = 0.30$	0.1564	0.3054	0.2590	0.1997	0.1518
element $\beta = 1.4, \lambda_e = 0.30$	0.1579	0.3065	0.2669	0.1993	0.1515
element $\beta = 1.5, \lambda_e = 0.30$	0.1567	0.3088	0.2650	0.1961	0.1508

the results for the All in Context Task. The official measure is the mean average generalized precision in column 2. 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.

Table 13 also evaluates the runs using the same measures as the Thorough and Focused Task above. When comparing the All in Context Task results to the Focused Task results, we see that the clustering by article improves performance for all measures. That is, clustering the elements per article is more effective than ranking them on their own similarity score. This is an unexpected result that may be related to the organization of information in an encyclopedia like Wikipedia.

Table 14 shows the results for non-official runs for the All in Context Task, using different values for the language model parameters. The runs on the element index are all inferior to the official run on the section index. Again, as may be expected, a higher value of λ_e leads to a small increase in precision. In fact, it also results in an increase of MAep. Moreover, a higher value of β also leads to an increase of both precision and MAep. A possible explanation is the generalized precision measure that treats "retrieved articles" as ranks, and the fact that there is usually only a small number of articles with relevance in the Wikipedia collection. This leads to an incentive to ensure that at least those articles are retrieved.

Table 15. Results for the Best in Context Task (generalized, off)

Run	A=0.01	A=0.1	A=1	A=10	A=100
`best_section_lm.highest_score`	0.1237	0.2103	0.3437	0.5365	0.7369
`best_section_lm.first`	0.1237	0.2103	0.3437	0.5365	0.7369
`best_section_lm.article`	0.0754	0.1761	0.3027	0.4853	0.7021

Table 16. 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	0.2083	0.1678	0.0131
`best_section_lm.first`	0.3290	0.2796	0.2083	0.1678	0.0131
`best_section_lm.article`	0.2451	0.1983	0.1424	0.1109	0.0085

8 Experiments for the Best in Context Task

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 post-process 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.

8.1 Results

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. With high A values the position of the Best-Entry-Point in the article has very little impact, leading to scores of 1 for a Best-Entry-Point anywhere in the document. Table 15 shows 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. As mentioned above, with higher A values the position of the BEP has less effect, which can be seen by the convergence of scores for the different methods. Our Best in Context runs where based on postprocessing the All in Context run, resulting 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.

In Table 16, we also evaluate runs here using the same measures as the Thorough Task above, so optimal performance will result in still grossly imperfect scores. When comparing the Best in Context Task results to the All in Context

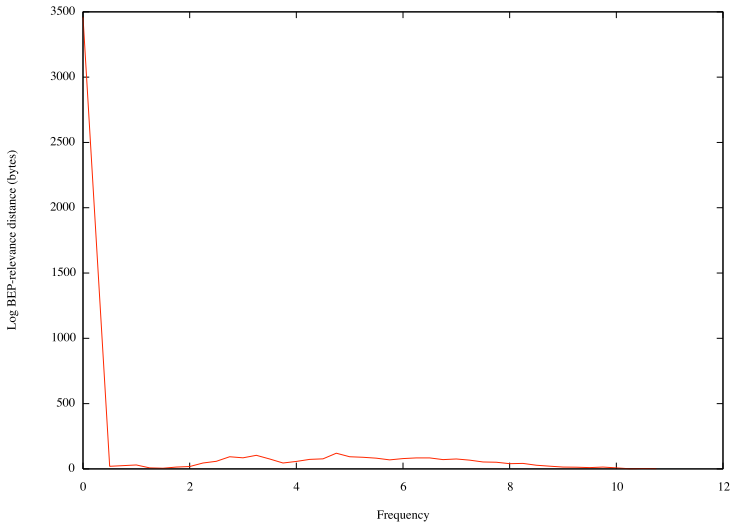

Fig. 2. Distance between BEP and first highlighted character

Table 17. Mixture model results for the Thorough Task (generalized, off)

Run	MAep	nxCG@5	nxCG@10	nxCG@25	nxCG@50
element $\lambda_e = 0.05, \lambda_d = 0.10$	0.0433	0.3429	0.3176	0.2776	0.2324
element $\lambda_e = 0.10, \lambda_d = 0.05$	0.0447	0.3640	0.3358	0.2918	0.2457
element $\lambda_e = 0.10, \lambda_d = 0.10$	0.0447	0.3693	0.3361	0.2939	0.2464
element $\lambda_e = 0.20, \lambda_d = 0.10$	0.0452	0.3895	0.3477	0.2976	0.2506
element $\lambda_e = 0.30, \lambda_d = 0.10$	0.0452	0.3900	0.3481	0.3041	0.2513

Task results, we note that there is only a moderate loss of precision for the Best in Context Task.

This result can be explained if we look at the position of the Best-Entry-Point relative to the first highlighted character in relevant articles. Figure 2 shows the distribution of absolute distance in bytes between the Best-Entry-Point and the position of the first highlighted character (i.e., the position at which the first passage starts) over articles with relevance. Clearly, assessors predominantly judge the first highlighted character to be the best point to start reading.

9 Mixture model runs

In this section we look at the performance of the multinomial language model that takes the context of an element into account. We experimented with various λ_e and λ_d values of the mixture language model and compare performance with the runs mentioned above.

Table 17 shows that less smoothing on the element model improves performance, not only for the official MAep measure, but also for precision at all levels. The difference between $\lambda_e = 0.20$ and $\lambda_e = 0.30$ is very small. If we compare

Table 18. Mixture model results for the Focused Task (generalized, off)

Run	nxCG@5	nxCG@10	nxCG@25	nxCG@50	MAep
element $\lambda_e = 0.05, \lambda_d = 0.10$	0.2987	0.2662	0.1997	0.1542	0.0116
element $\lambda_e = 0.10, \lambda_d = 0.05$	0.3219	0.2786	0.2134	0.1639	0.0125
element $\lambda_e = 0.10, \lambda_d = 0.10$	0.3220	0.2817	0.2136	0.1639	0.0124
element $\lambda_e = 0.20, \lambda_d = 0.10$	0.3334	0.2847	0.2174	0.1688	0.0127
element $\lambda_e = 0.30, \lambda_d = 0.10$	0.3335	0.2833	0.2214	0.1696	0.0130

Table 19. Mixture model results for the All In Context Task (generalized, off, highest scoring element)

Run	MAgP	gP@5	gP@10	gP@25	gP@50
element $\lambda_e = 0.05, \lambda_d = 0.10$	0.1403	0.2603	0.2254	0.1719	0.1319
element $\lambda_e = 0.10, \lambda_d = 0.05$	0.1486	0.2786	0.2389	0.1840	0.1397
element $\lambda_e = 0.10, \lambda_d = 0.10$	0.1489	0.2804	0.2416	0.1847	0.1405
element $\lambda_e = 0.20, \lambda_d = 0.10$	0.1530	0.2957	0.2517	0.1924	0.1472
element $\lambda_e = 0.30, \lambda_d = 0.10$	0.1538	0.2969	0.2540	0.1943	0.1480

the results with runs based on the element index and the section index (see Table 8), performance of the mixture model is comparable to that of the section index run. Also, setting the article model smoothing parameter λ_d higher also slightly improves precision scores. In sum, the mixture language model has a slight positive effect on precision, but does not improve overall performance.

Table 18 shows the results for the Focused Task. Here, the effects of the two smoothing parameters are very similar to the Thorough Task. When compared to the official element and section based runs (Table 10), we see no gain in performance.

Table 19 shows the results for the All In Context Task, where we restrict our attention to the article ordering based on the highest scoring element. The results show the same effect of the smoothing parameters as for the other tasks. A higher λ_e leads to better performance. The difference between $\lambda_e = 0.20$ and $\lambda_e = 0.30$ is very small again. The mixture model can lead to improvement of precision and MAgP, however the gain is inferior to the increase of length normalization (see Table 14).

The mixture language model proved far less effective for the Wikipedia collection, than earlier research on the IEEE collection [4]. An obvious source of explanation is in the relative length of Wikipedia articles: recall that the average article contains only 415 words. This makes the article context a less powerful indicator of relevance, and perhaps the model should be extended to incorporate the broader of the particular wikipedia page (for example by considering the rich, semantic link structure).

10 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, and conducted

a range of experiments with filtering and clustering XML element retrieval results. Our basic retrieval engine retrieves arbitrary elements from the collection (corresponding to the Thorough Task). These runs are filtered to remove textual overlap between elements (corresponding to the Focused Task). The resulting runs can be clustered per article (corresponding to the All in Context Task). Finally, we select the "best" element for each article (corresponding to the Best in Context Task).

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 surprisingly—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)
7. Sigurbjörnsson, B.: Focused Information Access using XML Element Retrieval. SIKS dissertation series 2006-28, University of Amsterdam (2006)
8. Kamps, J., de Rijke, M., Sigurbjörnsson, B.: The importance of length normalization for XML retrieval. Information Retrieval **8** (2005) 631–654