Relevance Feedback for Structural Query Expansion

Ralf Schenkel and Martin Theobald

Max-Planck-Institut für Informatik, Saarbrücken, Germany
{schenkel, mtb}@mpi-inf.mpg.de

Abstract. Keyword-based queries are an important means to retrieve information from XML collections with unknown or complex schemas. Relevance Feedback integrates relevance information provided by a user to enhance retrieval quality. For keyword-based XML queries, feedback engines usually generate an expanded keyword query from the content of elements marked as relevant or nonrelevant. This approach that is inspired by text-based IR completely ignores the semistructured nature of XML. This paper makes the important step from pure content-based to structural feedback. It presents two independent approaches that include structural dimensions in a feedback-driven query evaluation: The first approach reranks the result list of a keyword-based search engine, using structural features derived from results with known relevance. The second approach expands a keyword query into a full-fledged content-and-structure query with weighted conditions.

1 Introduction and Motivation

XML has seen increasing importance recently to represent large amounts of semistructured or textual information in digital libraries, intranets, or the Web, so information retrieval on XML data is growing more and more important. XML search engines employ the ranked retrieval paradigm for producing relevance-ordered result lists rather than merely using XPath or XQuery for Boolean retrieval. An important subset of XML search engines uses keyword-based queries [2, 7, 30], which is especially important for collections of documents with unknown or highly heterogeneous schemas. However, simple keyword queries cannot exploit the often rich annotations available in XML, so the results of an initial query are often not very satisfying.

Relevance Feedback is an important way to enhance retrieval quality by integrating relevance information provided by a user. In XML retrieval, existing feedback engines usually generate an expanded keyword query from the content of elements marked as relevant or nonrelevant. This approach that is inspired by text-based IR completely ignores the semistructured nature of XML.

This paper makes the important step from content-based to structural feedback. We present two independent approaches to exploit the structure of XML with relevance feedback:

1. Using the feedback approach by Rocchio [20], we create new content and structural features that are used to rerank the results of a keyword-based engine, enabling structural feedback for engines that support only keyword-based queries.
2. We extend the feedback approach by Robertson and Sparck-Jones [19] to expand a keyword-based query into a possibly complex content-and-structure query that specifies new constraints on the structure of results, in addition to "standard" content-based query expansion. The resulting expanded query has weighted structural and content constraints and can be fed into a full-fledged XML search engine like our own TopX [26] Search Engine.

2 Related Work

Relevance feedback has already been considered for document retrieval for a long time, starting with Rocchio's query expansion algorithm [20]. Ruthven and Lalmas [21] give an extensive overview about relevance feedback for unstructured data, including the assessment of relevance feedback algorithms.

Relevance feedback in XML IR is not yet that popular. Of the few papers that have considered it, most concentrate on query expansion based on the content of elements with known relevance [4, 13, 25, 29]. Some of these focus on blind ("pseudo") feedback, others on feedback provided by users. Pan et al. [15] apply user feedback to recompute similarities in the ontology used for query evaluation.

Even fewer papers have considered structural query expansion [8, 9, 14, 16, 17]. Mihajlovič et al. [14, 16, 17] proposed deriving the relevance of an element from its tag name, but could not show any significant gain in retrieval effectiveness. Additionally, they considered hand-tuned structural features specific for the INEX benchmark (e.g., the name of the journal to which an element's document belongs), but again without a significant positive effect. In a follow-up to this work, Ramírez et al. [17] could show significant improvements with journal names. In contrast, our general approach for exploiting feedback can be applied with the INEX data, but does not rely on any INEX-specific things.

Hlaoua and Boughanem [8] consider common prefixes of relevant element's paths as additional query constraints, but don't provide any experimental evaluation of their approach.

Gonçalvez et al. [5] use relevance feedback to construct a restricted class of structured queries (namely field-term pairs) on structured bibliographic data, using a Bayesian network for query evaluation. While they did not consider XML, their overall approach is somewhat similar to our reranking framework presented in Section 4.

The work of Hsu et al. [9] is closest to our approach. They use blind feedback to expand a keyword-based query with structural constraints derived from a neighborhood of elements that contain the keywords in the original query. In contrast, our approach considers the whole document instead of only a fragment, can generate constraints with negative weight, and integrates also content-based constraints.

3 Formal Model and Notation

3.1 Data Model

We consider a fixed corpus of D XML documents with their elements. For such an element e, $t(e)$ denotes its tag name and $d(e)$ the document to which it belongs.

The *content* $c(e)$ of an element e is the set of all terms (after stopword removal and optional stemming) in the textual content of the element itself and all its descendants. For each term t and element e, we maintain a weight $w_e(t)$. This can be a binary weight ($w_e(t) = 1$ if the term occurs in e's content and 0 otherwise), a tf-idf style [12] or a BM25-based [1, 28] weight that captures the importance of t in e's content. The *content* $c(d)$ of a document d is defined as the content $c(r)$ of its root element r.

We maintain a number of statistics about the occurrence of terms in documents and elements: The *document frequency* df_t of a term t is the number of documents in which the term appears in the content. Analogously, the *element frequency* ef_t of a term t is the number of elements in which the term appears in the content.

3.2 Queries and Relevance of Results

We use an extended version of INEX's query language NEXI [27]. NEXI basically corresponds to XPath restricted to the `descendants-or-self` and `self` axis and extended by an IR-style `about` predicate to specify conditions that relevant elements should fulfil. The wildcard symbol '*' matches any tag and can be used to formulate keyword queries in NEXI. We extend NEXI with additional weights for each content constraint. A typical extended NEXI query looks like the following:

```
//article[about(.,"0.8*XML")//*[about(.//p,"0.4*IR -0.2*index")]
```

The result granularity of such a query are elements. We currently assume that the relevance of an element with respect to a query is measured with the strict quantization, i.e., an element is either relevant or nonrelevant.

3.3 Feedback Model

We consider a keyword query $q = \{q_1, \ldots, q_p\}$ with a set $E = \{e_1, \ldots, e_l\}$ of results with known relevance. i.e., elements for which a user has assigned an exhaustivness value $e(e)$ and a specificity value $s(e)$. Using the strict quantization, we say that an element e is *relevant* for the query if both $e(e)$ and $s(e)$ are maximal, yielding a set $E^+ = \{e_1^+, \ldots, e_R^+\}$ of relevant elements and a set $E^- = \{e_1^-, \ldots, e_N^-\}$ of nonrelevant elements.

Note that even though we consider only binary relevance, it is possible to extend the mechanism presented here to approaches where relevance is measured with a probability-like number between 0 and 1, for example by representing E^+ and E^- as probabilistic sets.

4 Reranking Results of Keyword-Only Runs

Our first approach aims at identifying documents that contain relevant elements and paths of relevant elements, in addition to standard content-based query expansion. We first compute the results for a keyword query with an existing keyword-based engine and ask a user for relevance feedback. Based on the user input, we compute certain classes of *features* from elements with relevance feedback and select those that best discriminate relevant from nonrelevant results. Using these features, we compute additional scores for element-feature-matches for all remaining elements and rerank them by their combined score. This approach allows to evaluate certain classes of structural constraints with engines that support only keyword-based queries.

For space restrictions we can only informally present our approach here; a more detailed and formal description can be found in [23].

4.1 Features Used for Reranking

We derive the following classes of candidates for query expansion from an element with known relevance:

- all terms of the element's content together with their score (C features),
- tag-term pairs within the element's document (D features), and
- features derived from the path of the element (P features).

The system can be extended with additional classes of features.

Content Features. Content-based feedback is widely used in standard IR and has also made its way into XML retrieval [13, 25]. It expands the original query with new, weighted keywords that are derived from the content of elements with known relevance. As an example, consider the keyword query `"multimedia information" retrieval` (this is topic 178 from the INEX topic collection). From the feedback of a user, we may derive that elements that contain the terms 'brightness', 'annotation', or 'rgb' are likely to be relevant, whereas elements with 'hypermedia' or 'authoring' are often irrelevant.

Document Features. Unlike standard text retrieval where the unit of retrieval are whole documents, XML retrieval focuses on retrieving parts of documents, namely elements. Information in other parts of a document with a relevant element can help to characterize documents in which relevant elements occur. A natural kind of such information is the content of other elements in such documents.

As an example, consider again INEX topic 178 (`"multimedia information" retrieval`). We may derive from user feedback that documents with the terms 'pattern, analysis, machine, intelligence' in the journal title (i.e., those from the 'IEEE Transactions on Pattern Analysis and Machine Learing') are likely to contain relevant elements. The same may hold for documents that cite papers by Gorkani and Huang (who are co-authors of the central paper about the

QBIC system), whereas documents that cite papers with the term 'interface' in their title probably don't contain relevant elements (as they probably deal with interface issues in multimedia applications).

Other possible structural features include twigs, occurence of elements with certain names in a document, or combination of path fragments with terms. Further exploration of this diversity is subject to future work.

Path Features. Elements with certain tag names are more likely to be relevant than elements with other tag names. As an example, a keyword query may return entries from the index of a book or journal with high scores as they often contain exactly the requested keywords, but such elements are usually not relevant. Additionally, queries may prefer either large elements (such as whole articles) or small elements (such as single paragraphs), but rarely both. However, experiments show that tag names alone do not bear enough information to enhance retrieval quality, but the whole path of a result element plays an important role. As an example, the relevance of a paragraph may depend on whether it is in the body of an article (with a path like `/article/bdy/sec/p` from the root element), in the description of the vitae of the authors (with a path like `/article/bm/vt/p`), or in the copyright statement of the journal (with a path like `/article/fm/cr/p`).

As element tag names are too limited, but complete paths may be too strict, we consider the following six classes of *path fragments*, with complete paths and tag names being special cases:

- P_1: prefixes of paths, e.g., `article/#`,`/article/fm/#`
- P_2: infixes of paths, e.g., `#/fm/#`
- P_3: subpaths of length 2, e.g., `#/sec/p/#`
- P_4: paths with wildcards, e.g, `#/bm/#/p/#`
- P_5: suffixes of paths, e.g., `#/fig`, `#/article`
- P_6: full paths, e.g, `/article/bdy/sec`

Mihajloviĉ et al. [14] used a variant of P_5, namely tag names of result elements, but did not see any improvement. In fact, only a combination of fragments from several classes leads to enhancements in result quality. Experiments in [23] show that the best results are yielded with a combination of P1, P3 and P4, whereas using P5 or P6 alone actually reduced the quality of results below the baseline without feedback.

4.2 Feature Weights and Selection

We compute the weight for all features using the standard Rocchio weight [20]. We tried several variations, including binary and weighted Rocchio weights, and have yet to explore the whole space of solutions.

Among the (usually many) possible features, we choose the n_c content features and n_s document features with highest absolute weights. If there are too many with the same weight, we use the mutual information of the feature's score distribution among the elements with known relevance and the relevance distribution as a tie breaker. If there are no positive examples and mutual information is zero for all features, we use the feature's document frequency (the number of documents in which this feature occurs) for tie breaking then, preferring features that occur infrequently.

4.3 Reranking Results

For each element that occurs in the baseline run, we compute an additional score for each feature class. The score for each feature class is computed in a separate vector space where each dimension corresponds to a feature that occurs in at least one element. The score of the element for this class is then computed as the cosine of the vector with the selected features for this dimension and the element's feature vector. Each of the scores is normalized to the interval $[-1.0, 1.0]$. The overall score of the element is then the sum of its score from the baseline run and its additional scores.

This scoring model can easily integrate new dimensions for feedback beyond content, path and document features, even if they use a completely different model (like a probabilistic model). It only requires that the relevance of an element to a new feedback dimension can be measured with a score between -1 and 1. It is simple to map typical score functions to this interval by normalization and transformation. As an example, the transformation rule for a probability p, $0 \leq p \leq 1$, is $2 \cdot p - 1$.

5 Generating Structural Queries from Feedback

Keyword-based queries are the best way to pose queries without knowledge of the underlying schema of the data, but they cannot exploit the structure of documents. As an example, consider the keyword query (query 230 from the INEX benchmark [11]) `+brain research +"differential geometry"`, asking for applications of differential geometry in brain research. In relevant results, "brain research" is usually the topic of the whole article, while "differential geometry" is typically the topic of a section. A query with constraints on both content and structure would probably yield a lot more relevant results, but it is impossible to formulate a query like the following without knowledge of the underlying schema:

```
//article[about(.,brain research)]//sec[about(.,differential geometry)]
```

We studied the content-and-structure queries from INEX to find patterns that are regularily used in such queries to describe relevant elements, in addition to content conditions on the result element. A canonical example for such a query is the following:

```
//article[about(.,"RDF") and about(//bib,"W3C")]//sec[about(.,"query")
and about(//par,"performance")]
```

This is a content-and-structure version of the simpler keyword query "RDF W3C query performane". In contrast to the keyword query, the structured query specifies a tag (or, more generally, a set of tags) that relevant elements should have ("I am interested in sections about 'query'"). Additionally, this query contains constraints on the content of descendants of relevant elements ("sections with a paragraph about 'performance'"), the content of ancestors ("sections in articles about 'RDF'"), and the content of descendants of ancestors ("sections in articles that cite a paper from the 'W3C'").

As such a content-and-structure query specifies much more precisely the conditions that relevant elements must satisfy, we can expect that a search engine will return more relevant results for a content-and-structure query than for the keyword query, provided that the content-and-structure query correctly captures the same information need as the keyword query. Our feedback framework aims at generating a content-and-structure query from a keyword query, exploiting relevance feedback provided by a user for some results of the keyword query.

For space restrictions we can only informally present our approach here; a more detailed and formal description can be found in [24].

5.1 Candidates for Query Expansion

Following the discussion in the beginning of this section, we derive the following classes of candidates for query expansion from an element with known relevance:

– all terms of the element's content together with their score (C candidates),
– all tag-term pairs of descendants of the element in its document, together with their score (D candidates),
– all tag-term pairs of ancestors of the element in its document, together with their score (A candidates), and
– all tag-term pairs of descendants of ancestors of the element in its document, together with their score and the ancestor's tag (AD candidates).

The system can be extended with additional classes of candidates like tags, twigs, or paths, which is subject to future work.

The candidate set of an element is the set of all possible candidates for this element. We extend the notion of frequencies from terms to candidates as follows: the element frequency of a candidate is the number of elements where the candidate appears in the candidate set, and its document frequency is the number of documents with at least one element with the candidate in their candidate set.

5.2 Candidate Weights and Selection

To weight the different candidates, we apply an extension of the well-known Robertson-Sparck-Jones weight [19] to element-level retrieval in XML, applying it for elements instead of documents:

$$w_{RSJ}(c) = \log \frac{r_c + 0.5}{R - r_c + 0.5} + \log \frac{E - ef_c - R + r_c + 0.5}{ef_c - r_c + 0.5}$$

Here, for a candidate c, r_c denotes the number of relevant elements which contain the candidate c in their candidate set, R denotes the number of relevant elements,

E the number of elements in the collection, and ef_c the element frequency of the candidate.

The set of all possible expansion candidates is usually very large and contains many unimportant and misleading expansions, so we have to select the best b of them for generating the expanded query. This problem already exists for content-based expansion of keyword queries, and several possible weights have been proposed in the literature that go beyond naively ordering terms by their weight. We use the so-called Robertson Selection Values (RSV) proposed by Robertson [18]. For a candidate c, its RSV has the form $RSV(c) = w_{RSJ}(c) \cdot (p - q)$, where $p = r_c/R$ is the estimated probability of the candidate occurring in a relevant element's candidate set and q is the probability that it occurs in a nonrelevant element's set. We ignore candidates that occur only within the documents of elements with known relevance as they have no potential to generate more relevant results outside these documents. We order the union of the remaining candidates by their RSV and choose the top b of them, where b is a configuration parameter of the system. To be able to generate a valid NEXI query in the next step, we have to limit the A and AD candidates chosen to contain the same ancestor tag.

5.3 Generating an Expanded Query

Using the top-b candidates, we generate a content-and-structure query from the original keyword query. This expansion is actually straight-forward, and the generated query has the following general structure:

```
//ancestor-tag[A+AD constraints]//*[keywords+C+D constraints]
```

As an example, if the original query was 'XML' and we selected the A candidate (anc,article,'IR'), the AD candidate (article,bib,'index') and the D candidate (desc,p,'index'), the expanded query would be

```
//article[about(.,'IR') and about(//bib,'index')]//*[about(.,'XML')
and about(//p,'index')]
```

Each of the expansions is weighted, where the weight is the candidate's RSJ weight adjusted by a factor that depends on the candidate's class. C and D candidates help finding new relevant results, so they should get a high weight; we allow for C and D conditions at most the weight of all original keywords (to make sure that the new constraints don't dominate the query's results). As an example, for a query with four keywords and six C and D expansions, the factor for each expansion is $\frac{4}{6}$. On the other hand, A and AD conditions are satisfied by most – if not all – elements of a document, so they generate a huge amount of new result elements, most of which will be nonrelevant. Their weight should therefore be smaller than the weight of C and D conditions. We choose a fraction β of the accumulated weight of existing keyword conditions, with $\beta = 0.2$ in our experiments.

6 Architecture and Implementation

We have implemented the reranking approach from Section 4 and the query expansion approach from Section 5 within an automated system that can import queries and results from INEX and automatically generate feedback for the top-k results, using the existing INEX assessments.

Our Java-based implementation requires that important information about elements is precomputed: unique identifiers for the element (`eid`) and its document (`did`), its pre and post order to facilitate the evaluation of structural query conditions like the XPath axes [6] or any other similar information, its tag, and its terms (after stemming and stopword removal), together with their score. This information is stored in a database table with schema (<u>`did`</u>,<u>`eid`</u>,<u>`term`</u>,`tag`,`pre`,`post`,`score`) that contains one tuple for each distinct term of an element. Our current prototype reuses the `TagTermFeatures` table of TopX (see [26]) that already provides this information. On the database side, we provide indexes on `(eid,did)` to efficiently find $d(e)$ for an element e and on `(did)` to efficiently collect all elements of a document. Inverse element and document frequencies of the different candidate classes are precomputed (e.g., while initially parsing the collection) and stored in database tables, too.

7 Evaluation of Feedback Runs

The evaluation of feedback runs for XML IR is a problem that has not yet been solved in a satisfying way. People have agreed that simply comparing the results of a run with feedback to the baseline run (which we call *plain* later) is unfair as the new run has information about relevant elements and hence is biased. The INEX Relevance Feedback track has used two different measures so far:

– In 2004, a variant of the residual collection technique was proposed [1]. Here, all XML elements with known relevance must be removed from the collection before evaluation of the results with feedback takes place. This means not only each element used or observed in the RF process but also all descendants of that element must be removed from the collection (i.e., the residual collection, against which the feedback query is evaluated, must contain no descendant of that element). All ancestors of that element are retained in the residual collection.
– In 2005, the rank of results with know relevance is frozen, thus assessing only the effect of reranking the results with unknown relevance. We label this approach *freezeTop* as usually the top-k results are used for feedback and hence frozen.

Using the residual collection approach opens up a variety of different evaluation techniques:

– *resColl-result*: only the elements for which feedback is given are removed from the collection,
– *resColl-desc*: the elements for which feedback is given and all their descendants are removed from the collection (this is the technique used in this year's RF track),

[1] This is stated on the official homepage of the INEX 2004 RF track [10]. We do not know to which extend it was really used by the participants.

- *resColl-anc*: the elements for which feedback is given and all their ancestors are removed from the collection, and
- *resColl-doc*: for each element for which feedback is given, the whole document is removed from the collection.

We evaluate our approaches with all six evaluation techniques in the following section and try to find out if there are any anomalies.

8 Experimental Results

8.1 Official Runs

We submitted reranking runs for the best CO.Thorough runs of our TopX engine [26] (`MPII_TopX_CO.Thorough_exppred`) and our XXL engine [22] (`XXL++; CO.Thorough;andish;noExp`), using 5 C and 5 D features. Additionally, we submitted naive reranking runs that simply boost the score of elements within documents with at least one known relevant element (from the feedback). For the structural query expansion technique, we submitted only a run based on the TopX baseline run (this approach is implemented only on top of TopX), using the best 10 of all expansion candidates. For the official runs, the algorithms were provided with the relevance (exhaustivity and specificity) of the top 20 results of the baseline run. Using the *freeze-top* approach, these results with known relevance are replicated at the top of the newly generated runs. The evaluation measured absolute and relative gain over the corresponding baseline runs for the nxCG[50] and ep/gr metrics with the strict, generalized and generalisedLifted quantizations.

For all evaluations, the reranking runs outperformed the query expansion approach and most (usually all) runs submitted by other participants, with peak relative gains of 335% (for the TopX-based non-naive reranking with ep/gr and strict). The naive reranking runs did surprisingly well, with peak relative gains of 196% for the same metrics and quantization. The query expansion approach consistently gave less improvements than the reranking approaches; our experiments with the old inex_eval metric (see below) yielded the opposite result, so this needs further exploration.

8.2 Additional Runs

For the additional unofficial runs that were created after the official deadline, we used a CO.Thorough run created with an enhanced version of TopX engine for the 2005 CO topics[2]. Table 1 shows the macro-averaged precision for this run for the top-k ranked elements per topic, for different k; this corresponds to the average fraction of relevant results among the elements used for top-k feedback.

[2] The main reason for the relatively low MAP value of this run compared to our official run is the result of topic 228 where the new run ranks the only result in rank 2 instead of 1, yielding a MAP difference for this topic of about 0.7.

Table 1. Precision at k for the baseline run

k	10	15	20
prec@k	0.0593	0.0519	0.0500

Note that the average precision is much lower than in previous experiments with the INEX 2003 and 2004 CO topics where TopX yielded an average precision of 0.231 at the top-5 and still 0.174 at the top-20 results. We can effectively use only 10 of the 40 2005 CO topics (those with assessments and with at least one relevant result among the top-20 results) for the experiments, which makes the significance of the experiments at least questionable and at the same time explains the relatively low improvements shown in the following subsections. Table 2 gives some more information on this issue: it shows the number of topics in the baseline run that have a certain number of relevant results among the top k, for varying k. Again, it is evident that most topics do not have any relevant results at all. As an additional problem, there are some topics with only a few relevant results, which makes possible that slight changes in the order of results cause huge differences in the resulting MAP values. We decided therefore to present only our results with top-20 feedback in the following as the other results would not be significant anyway.

Table 2. Number of topics in the baseline run with r relevant results among the top k results

k/r	0	1	2	3	4	5	6
5	23	1	3	0	0	0	-
10	19	5	0	2	0	1	0
15	18	5	0	2	1	0	1
20	17	4	1	3	0	0	2

For each run, we measured the MAP using inex_eval with the strict quantization and the latest set of assessments. We plan to evaluate our results with other metrics in the future.

Results for Reranking Queries. Table 3 shows the MAP values of our experiments with different combinations of features to rerank the initial results, providing relevance feedback for a different number of top elements of the baseline run and selecting the best 5 features of each dimension, for the different evaluation techniques.

The results are surprisingly different from our earlier results in [23]: the absolute improvements are quite small, and the best results are achieved with either only P, only D, or both candidates. We attribute this to the fact that there are at most 10 topics with relevant results in the top-k; the remaining topics provide only negative feedback which seems to be not helpful here. Additionally, it

Table 3. MAP values for top-20 feedback runs with different configurations and different evaluation methods, for the reranking approach

evaluation	baseline	C	P	C+P	D	C+D	D+P	C+D+P
plain	0.0367	0.0465	0.1008	0.0534	0.0911	0.0492	**0.1120**	0.0563
resColl-result	0.0262	0.0343	**0.0581**	0.0216	0.0412	0.0312	0.0579	0.0228
resColl-anc	0.0267	0.0340	0.0581	0.0198	0.0400	0.0297	**0.0589**	0.0219
resColl-desc	0.0330	0.0180	0.0489	0.0142	0.0284	0.0132	**0.0498**	0.0151
resColl-doc	0.0309	0.0140	**0.0480**	0.0114	0.0249	0.0097	0.0468	0.0126
freezeTop	0.0367	0.0367	0.0371	0.0353	**0.0373**	0.0369	0.0362	0.0358

is evident that the different evaluation methods don't agree at all about which combination gives the best results. Other than that, it is interesting that some runs (like the D run which looks like the absolutely best choice with the freezeTop evaluation) do great with some evaluation techniques, but perform worse than the baseline with others. This is an anomaly that should be investigated further.

Results for Queries with Structural Constraints. Table 4 shows a comparison of MAP values with the different evaluation techniques of our experiments with different combinations of candidate classes for query expansion, providing relevance feedback for the top 20 elements of the baseline run and selecting the best 10 candidates for expansion.

The results are less impressive than we had expected after earlier experiments with the 2003 and 2004 CO topics. The best of our techniques (which consistently is the combination of all candidate classes for all five evaluation techniques, not counting the plain run) yields a performance gain of about 5%–20%, whereas we could show up to 100% in the other experiments (with resColl-desc). We think that there are mainly two reasons for this difference: The 2005 topics are inherently more difficult than the older topics (which is also reflected in the much lower MAP scores for the best runs this year), and there are only 10 topics where our approaches have a chance to enhance the quality (because Robertson-Sparck-Jones weights are not useful without relevant results) with top-20 feedback and

Table 4. MAP values for top-20 feedback runs with different configurations and different evaluation methods, for the query expansion approach

evaluation	baseline	C	D	C+D	A	AD	A+AD	A+C+D+AD
plain	0.0367	0.0419	**0.0707**	0.0406	0.0646	0.0663	0.0654	0.0513
resColl-result	0.0262	0.0294	0.0300	0.0295	0.0309	0.0306	0.0294	**0.0324**
resColl-anc	0.0267	0.0350	0.0356	0.0305	0.0298	0.0294	0.0296	**0.0366**
resColl-desc	0.0330	0.0353	0.0355	0.0355	0.0363	0.353	0.0346	**0.0365**
resColl-doc	0.0309	0.0317	0.0313	0.0314	0.0321	0.0310	0.0315	**0.0325**
freezeTop	0.0367	0.0378	0.0374	0.0375	0.0384	0.0378	0.0380	**0.0387**

even fewer for the other runs. Absolute values are slightly higher than the values achieved with the reranking approach, so choosing the best candidates out of a pool of expansion candidates instead of picking a fixed number from each class seems to perform better.

9 Conclusion and Future Work

This paper has made important steps from content-based to structural feedback in XML retrieval. It presented two independent approaches that exploit structural and content features: reranking an existing list of results and evaluating a structurally expanded query. The methods achieved good results in this year's Relevance Feedback Track and reasonable results for the earlier years.

Our future work will contentrate on more complex structural aspects of documents (like twigs, tags and paths) and extending this work to queries with content and structural constraints.

References

1. G. Amati, C. Carpineto, and G. Romano. Merging XML indices. In *INEX Workshop 2004*, pages 77–81, 2004.
2. A. Balmin et al. A system for keyword proximity search on XML databases. In *VLDB 2003*, pages 1069–1072, 2003.
3. H. Blanken, T. Grabs, H.-J. Schek, R. Schenkel, and G. Weikum, editors. *Intelligent Search on XML Data*, volume 2818 of *LNCS*. Springer, Sept. 2003.
4. C. J. Crouch, A. Mahajan, and A. Bellamkonda. Flexible XML retrieval based on the extended vector model. In *INEX 2004 Workshop*, pages 149–153, 2004.
5. M. A. Gonçalves, E. A. Fox, A. Krowne, P. Calado, A. H. F. Laender, A. S. da Silva, and B. Ribeiro-Neto. The effectiveness of automatically structured queries in digital libraries. In *4th ACM/IEEE-CS joint conference on Digital libraries (JCDL04)*, pages 98–107, 2004.
6. T. Grust. Accelerating XPath location steps. In *SIGMOD 2002*, pages 109–120, 2002.
7. L. Guo et al. XRANK: ranked keyword search over XML documents. In *SIGMOD 2003*, pages 16–27, 2003.
8. L. Hlaoua and M. Boughanem. Towards context and structural relevance feedback in XML retrieval. In *Workshop on Open Source Web Information Retrieval (OSWIR)*, 2005. `http://www.emse.fr/OSWIR05/`.
9. W. Hsu, M. L. Lee, and X. Wu. Path-augmented keyword search for XML documents. In *ICTAI 2004*, pages 526–530, 2004.
10. INEX relevance feedback track. `http://inex.is.informatik.uni-duisburg.de:2004/tracks/rel/`.
11. G. Kazai et al. The INEX evaluation initiative. In Blanken et al. [3], pages 279–293.
12. S. Liu, Q. Zou, and W. Chu. Configurable indexing and ranking for XML information retrieval. In *SIGIR 2004*, pages 88–95, 2004.
13. Y. Mass and M. Mandelbrod. Relevance feedback for XML retrieval. In *INEX 2004 Workshop*, pages 154–157, 2004.
14. V. Mihajlovič et al. TIJAH at INEX 2004 modeling phrases and relevance feedback. In *INEX 2004 Workshop*, pages 141–148, 2004.

15. H. Pan, A. Theobald, and R. Schenkel. Query refinement by relevance feedback in an XML retrieval system. In *ER 2004*, pages 854–855, 2004.
16. G. Ramirez, T. Westerveld, and A. de Vries. Structural features in content oriented xml retrieval. Technical Report INS-E0508, CWI,Centre for Mathematics and Computer Science, 2005.
17. G. Ramírez, T. Westerveld, and A. P. de Vries. Structural features in content oriented XML retrieval. In *CIKM 2005*, 2005.
18. S. Robertson. On term selection for query expansion. *Journal of Documentation*, 46:359–364, Dec. 1990.
19. S. Robertson and K. Sparck-Jones. Relevance weighting of search terms. *Journal of the American Society of Information Science*, 27:129–146, May–June 1976.
20. J. Rocchio Jr. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, chapter 14, pages 313–323. Prentice Hall, Englewood Cliffs, New Jersey, USA, 1971.
21. I. Ruthven and M. Lalmas. A survey on the use of relevance feedback for information access systems. *Knowledge Engineering Review*, 18(1), 2003.
22. R. Schenkel, A. Theobald, and G. Weikum. Semantic similarity search on semistructured data with the XXL search engine. *Information Retrieval*, 8(4):521–545, December 2005.
23. R. Schenkel and M. Theobald. Feedback-driven structural query expansion for ranked retrieval of XML data. In *10th International Conference on Extending Database Technologies (EDBT 2006)*, Munich, Germany, Mar. 2006.
24. R. Schenkel and M. Theobald. Structural feedback for keyword-based XML retrieval. In *28th European Conference on Information Retrieval (ECIR 2006)*, London, UK, Apr. 2006.
25. B. Sigurbjörnsson, J. Kamps, and M. de Rijke. The University of Amsterdam at INEX 2004. In *INEX 2004 Workshop*, pages 104–109, 2004.
26. M. Theobald, R. Schenkel, and G. Weikum. An efficient and versatile query engine for TopX search. In *VLDB 2005*, pages 625–636, 2005.
27. A. Trotman and B. Sigurbjörnsson. Narrowed Extended XPath I (NEXI). available at `http://www.cs.otago.ac.nz/postgrads/andrew/2004-4.pdf`, 2004.
28. J.-N. Vittaut, B. Piwowarski, and P. Gallinari. An algebra for structured queries in bayesian networks. In *INEX Workshop 2004*, pages 58–64, 2004.
29. R. Weber. Using relevance feedback in XML retrieval. In Blanken et al. [3], pages 133–143.
30. Y. Xu and Y. Papakonstantinou. Efficient keyword search for smallest LCAs in XML databases. In *SIGMOD 2005*, pages 537–538, 2005.