Exploring Path Query Results through Relevance Feedback

Huiping Cao, Yan Qi, K. Selçuk Candan
Arizona State Univ.
Tempe, AZ 85283, USA
{hcao11,yan.qi,candan}@asu.edu

Maria Luisa Sapino
Univ. di Torino
Torino, Italy
mlsapino@di.unito.it

ABSTRACT

Feedback driven data exploration schemes have been implemented for non-structured data (such as text) and document-centric XML collections where formulating precise queries is often impossible. In this paper, we study the problem of enabling exploratory access, through ranking, to data-centric XML. Given a path query and a set of results identified by the system to this query over the data, we consider feedback which captures the user's preference for some features over the others. The feedback can be "positive" or "negative". To deal with feedback, we develop a probabilistic feature significance measure and describe how to use this for ranking results in the presence of dependencies between the path features. We bring together these techniques in *AXP*, a system for *adaptive* and *exploratory path* retrieval. The experimental results show the effectiveness of the proposed techniques.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval: Relevance feedback.

General Terms

Algorithms, Measurement.

Keywords

Relevance feedback, inter-dependent structural feature, feature cover, data-centric XML.

1. INTRODUCTION

Feedback-based exploration of data sets has been shown to be very effective in non-structured data domains, such as text retrieval [7] and multimedia where formulating precise queries is often impossible. In the context of information retrieval (IR) and multimedia applications, where the data and queries on text collections are often modeled as keyword vectors [8], user feedback about the relevance or irrelevance of the result documents is used for adjusting the weights in the query vector by increasing the weights of the more relevant keywords [8].

In environments where the user may not know the underlying structure of the data which she is querying (thus is not able to form a precisely specified statements of interest), queries over structured and semi-structured data (e.g., tree or graph) may also suffer from similar problems. In particular, an incompletely specified query may return too many results. Consider, for example, the data in Figure 1, which describes several metadata items in a scientific repository; A query "*I am interested in the 'titles' in the current repository*" (i.e. $//title$), would return five matches. While there are some initial attempts, such as [5, 9], to address the challenge of enabling feedback on structured data (e.g., document-centric XML), this is still a largely unexplored area, with no effective strategies to support user feedback. One reason for this is that, while there are plenty of works on feedback on data that can be represented as vectors, not all data and queries are easy to map onto a vector space. This is especially true in semi-structured and structured data (e.g., data-centric XML documents, graphs), where the *structure* is often a critical component of the data. Path expressions – which express the desired characteristics of the paths on the data graph– combine requirements about values (such as the tags of an XML document) with requirements about the structural organization of these values. This renders the traditional feedback techniques inapplicable.

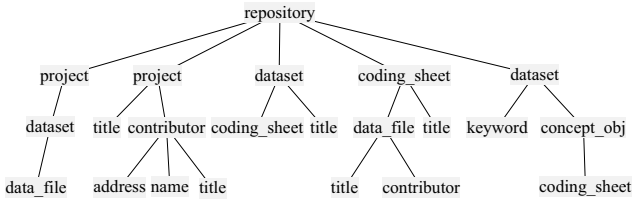

Figure 1: A small tree-structured data fragment

In this paper, we focus on the problem of feedback-based exploration of path results over data-centric XML. Such path results can be got from a path query. We recognize that, in some cases, the user is exploring the data within the context of an initial query but does not have well-defined correctness criteria in mind yet. In such cases, she may want the system to rank the results in the next iteration according to the positive or negative feedback she provides on the current results. To accommodate such declarations of preference, the system needs to support *feedback* and *ranking*. We propose an *adaptive, and exploratory, path retrieval (*AXP*)* framework to help a user explore the path results of her query by incorporating her preferences through positive and negative feedback.

1.1 Related Work

Query refinement through relevance feedback has been well studied in the fields of information retrieval [7] and multimedia. Most of the existing algorithms rely on the *vector model* [8] of data to support feedback-based query refinement. Intuitively, the vector describes the composition of the data or the query in terms of its constituent features (such as color, edge, or keyword). The vector model is especially suitable for supporting feedback, because the

CIKM'09, November 2–6, 2009, Hong Kong, China.

user feedback can be used both (a) to move the initial query vector in the vector space in a way that better represents the user's intentions or (b) to re-assess the significance of the features so that the query better reflects user's feedback.

As we described in the introduction, relevance feedback may also be needed when dealing with structured and semi-structured data. While vector models are proved to be effective in supporting feedback, not all data can be easily mapped onto a feature space and this is especially true for data with complex structures. The increasingly common usage of graph- and tree-structured data, such as XML documents, poses new challenges in data retrieval and feedback processing. A recent Dagstuhl report [3] on "Ranked XML Querying" highlights the significance of the "too-many-answers" problem to keyword-based XML retrieval. Observing that the traditional IR-based feedback processes cannot be directly applied to improve semi-structured data or XML retrieval, recent works, including [10, 6, 9, 5], have explored the role of relevance feedback for refining queries over XML documents. However, the existing work [6, 9, 5] focus on exploring document-centric INEX data set, but not data-centric XML collections. In addition, different from the above works, which only tackle positive feedback, the algorithms we present in the paper consider both positive and negative feedback.

2. PROBLEM SPECIFICATION

As mentioned in Section 1, when a path query returns a large number of results, it may not be feasible for the users to view and interpret all the results. To avoid such overload, given a large number, N, of paths, AXP presents the user only a small number, K, of these paths. Initially, these K paths are randomly selected and there is no specific presentation order. By providing feedback over the presented paths, the user is able to explore the alternative paths and focus on the subset that is of interest to her.

Data and query models. We focus on data-centric XML collections and path expressions of type $P^{\{/,//,*\}}$. The path expressions are composed of query steps, each consisting of an axis (parent/child "/" or ancestor/descendant "//") test and a label test. Given a document, T, and a path query, Q, the result to the query is an *embedding* of the query nodes onto the nodes of the data tree in such a way that all label and structural predicates are satisfied.

Problem of exploring path results. Let R be a set of path results of a query Q against a document collection T. Our goal is to improve the top K results (in R) by incorporating users' preferences (through relevance feedback) in an exploratory manner.

Structural features of a label path. The set, $\mathcal{F}(P)$, of features of a label-path $P = l_0 \cdot \cdots \cdot l_j$ is the maximal set of path queries such that $\mathcal{F}(P) = \{F_i \mid F_i(P) = true\}$ where F_i is in the type of $P^{\{/,//\}}$. A feature of a label-path can be as short as a 1-label like $//l_1$, or as long as a whole result path.

EXAMPLE 2.1. *Given a label-path $P = repository \cdot project \cdot title$, where "repository" is the label of the root node, $\mathcal{F}(P)$ includes single label features, $/project$, $//project$, $//title$, as well as 2-label features, $/project/title$, $/project//title$, $//project/title$, and $//project//title$ (we ignore the root label).*

Feedback model. *AXP* handles different forms of feedback: Given $r \in R$, let $P(r)$ be a data path defined by r. The user feedback over r is a 2-tuple $F = \langle F_s^+, F_s^- \rangle$:

- F_s^+ is a set of features of $P(r)$, which the user marks as positive feedback: the intended meaning is that the user LIKEs those results that contain these features better than others that do not contain these features.
- F_s^- is a set of features of $P(r)$, which the user marks as negative feedback: the intended meaning is that the user DISLIKEs any result containing the features included in F_s^-.

EXAMPLE 2.2. *Consider the path query "$//title$", when presented with the set of results the user might want to express that she would prefer to see dataset in the path results. Her feedback can be represented as the 2-tuple $\langle \{//dataset\}, \{\} \rangle$.*

Contributions: (1) We propose and formalize two kinds of feedback (LIKE and DISLIKE) to reorder path results over data-centric XML. (2)We propose probabilistic measures and algorithms for structural feature significance analysis by taking into account the structural dependencies between path features. We bring together these techniques in an *adaptive, and exploratory, path retrieval (*AXP*)* framework.

3. ENABLING PATH FEEDBACK

Given a set R of path results, the feedback is used for ranking the alternative results so that user can explore relevant parts of the data. *AXP* splits any complex feedback statement into its constituent 1- and 2-label feature elements for further analysis. These features capture the two key structural building blocks of label paths: 1-label features of the form "$//l_n$" capture existence of nodes with a certain label and 2-label features of the form "$//l_n/l_m$", and "$//l_n//l_m$" capture the parent/child and ancestor/descendant relationships between these labels. Note that, while they are simpler than more complex structural features, even these are not fully independent from each other; e.g., "$//l_n/l_m$" implies "$//l_n//l_m$". If a set of features are not mutually independent (e.g., one path feature includes the other one as a sub-feature), this might negatively affect the relevance feedback process: feature independence assumption is often necessary for simplifying the probabilistic formulas [7]. In Section 3.2.1, we describe how *AXP* addresses inter-dependences between the features.

3.1 Structural Feature Significance

Path features used during the feedback process need to distinguish those paths that are aligned with the user's relevance judgment from those that are against it. Thus, features can be weighted based on how well they contribute to this relevance judgment.

DEFINITION 3.1 (FEATURE SIGNIFICANCE). *Let R_K be the set of K results presented to the user for feedback, and F_s be the set of 1- or 2-label features extracted from the user's (LIKE or DISLIKE) relevance feedback statements. The* significance *of feature $f \in F_s$, relative to the result set, R_K denoted as $p(f|R_K, F_s)$ is defined as $p(f|R_K, F_s) = \frac{|R_K(f)|}{|R_K(F_s)|}$, where $R_K(f)$ is the set of results that satisfy the feature f, whereas $R_K(F_s)$ is the set of results that satisfy any of the feedback statements in F_s.*

EXAMPLE 3.1. *Consider two results to query, '$//title$': $r_1 = repository \cdot coding\_sheet \cdot title$, and $r_2 = repository \cdot coding\_sheet \cdot data\_file \cdot title$. Assume that the user marks '$/coding\_sheet$' in r_1 and '$//coding\_sheet/data\_file$' in r_2 as feedback. Thus, $F_s = \{/coding\_sheet, //coding\_sheet/data\_file\}$, $R_K = R_K(F_s) = \{r_1, r_2\}$. The 1-label feature '$f_1 = /coding\_sheet$' appears in both r_1 and r_2; thus, $p(f_1|R_K, F_s) = 1.0$. The 2-label feature '$f_2 = //coding\_sheet/data\_file$' is only in r_2; thus, $p(f_2|R_K, F_s) = 0.5$.*

3.2 Combined Result Score

We define the score of a result in terms of the significances of its features as they relate to the user feedback: If a result contains features that are significant relative to the positive feedback, intuitively, its score needs to be high; on the other hand, if the result

path contains features that are significant relative to the negative feedback, its overall score needs to be low. Let $\mathcal{F}_s^+$ and $\mathcal{F}_s^-$ represent positive and negative feedback, respectively.

DEFINITION 3.2 (RESULT SCORE). *The combined* score *of a result r_i is defined as follows:*

$$score(r_i) = \frac{p(Like|r_i, \mathcal{F}_s^+) \cdot p(Like|r_i, \mathcal{F}_s^-)}{p(Dislike|r_i, \mathcal{F}_s^-) \cdot p(Dislike|r_i, \mathcal{F}_s^+)}.$$

Here, $p(Like|r_i, \mathcal{F}_s^+)$ denotes the probability that result r_i will be judged relevant by the system based on the feedback in $\mathcal{F}_s^+$. Let R_K be the set of K results shown to the user for feedback and R_K^+ denote those that satisfy at least one of the positive feedback statements in $\mathcal{F}_s^+$. Then,

$$p(Like|r_i, \mathcal{F}_s^+) = \frac{p(r_i|R_K^+, \mathcal{F}_s^+)p(R_K^+, \mathcal{F}_s^+)}{p(r_i, \mathcal{F}_s^+)}.$$

On the other hand, $p(Dislike|r_i, \mathcal{F}_s^+)$ denotes the probability that result r_i is judged *not relevant* based on $\mathcal{F}_s^+$. Let $R_K^{\neg(+)} = R_K - R_K^+$ denote those results that do not satisfy any of the positive feedback statements in $\mathcal{F}_s^+$. Then,

$$p(Dislike|r_i, \mathcal{F}_s^+) = \frac{p(r_i|R_K^{\neg(+)}, \mathcal{F}_s^+)p(R_K^{\neg(+)}, \mathcal{F}_s^+)}{p(r_i, \mathcal{F}_s^+)}.$$

The terms, $p(Dislike|r_i, \mathcal{F}_s^-)$ and $p(Like|r_i, \mathcal{F}_s^-)$ are defined similarly based on $\mathcal{F}_s^-$. Note that, since they are computed based on different sets of feedback, $p(Dislike|r_i, \mathcal{F}_s^+)$ is not necessarily equivalent to $p(Dislike|r_i, \mathcal{F}_s^-)$. Similarly, for $p(Like|r_i, \mathcal{F}_s^-)$ and $p(Like|r_i, \mathcal{F}_s^+)$. Given these, and using the Bayes' theorem, we can rewrite $score(r_i)$ as follows:

$$\frac{p(r_i|R_K^+, \mathcal{F}_s^+) \cdot p(r_i|R_K^{\neg(-)}, \mathcal{F}_s^-)}{p(r_i|R_K^-, \mathcal{F}_s^-) \cdot p(r_i|R_K^{\neg(+)}, \mathcal{F}_s^+)} \times \frac{p(R_K^+, \mathcal{F}_s^+) \cdot p(R_K^{\neg(-)}, \mathcal{F}_s^-)}{p(R_K^-, \mathcal{F}_s^-) \cdot p(R_K^{\neg(+)}, \mathcal{F}_s^+)}.$$

Since the term at the right is constant, we only need to compute the term on the left. Given a feature set F' and a corresponding $R' \subseteq R_K$, we need to compute $p(r_i|R', F')$. Let $F'(r_i)$ denote the set of corresponding features of the path r_i. Since r_i is the conjunction of all of its features, we can write $p(r_i|R', F')$ as $p(\bigwedge_{f_j \in F'(r_i)} f_j|R', F')$. If all the features of r_i were independent, we could easily rewrite this as

$$p(r_i|R', F') = \prod_{f_j \in F'(r_i)} p(f_j|R', F') = \prod_{f_j \in F'(r_i)} \frac{|R'(f_j)|}{|R'|};$$

however, the 1- or 2-label features of the form "$//l_n$", "$//l_n/l_m$", and "$//l_n//l_m$" are not always independent. For example, feature "$//l_n//l_m$" implies the feature "$//l_n$". Similarly, "$//l_n/l_m$" implies "$//l_n//l_m$". Thus, such dependencies between features have to be taken into account in assessing their contributions.

3.2.1 Feature Cover

Given two inter-dependent features of a path, we refer to the one that always implies the other as the more specific feature:

DEFINITION 3.3 (SPECIFICITY). *Given two features f_g, f_s, the feature f_s is more* specific *than f_g, (denoted as $f_s \ll f_g$) iff for any label path P, f_g is a feature of P if f_s is a feature of P.*

Whenever the feature $//project/title$ is in a path, the feature $//project//title$ is also satisfied; thus $//project/title$ is a more specific feature than $//project//title$. Given a set of features, its *feature cover* consists of the most specific features in the set:

DEFINITION 3.4 (FEATURE COVER). *A feature cover F_c corresponding to a feature set F is the set of all features in F such that $\forall f_i \in F_c$, $\nexists f_j \in F$ s.t. $f_j \ll f_i$.*

THEOREM 3.1. *Let F be a set of features and F_c be the corresponding feature cover. Then, $p\left(\bigwedge_{f \in F} f\right) = p\left(\bigwedge_{f \in F_c} f\right)$.*

That is, the probability that any given path P contains all features in F is equal to the probability that the path contains all features in the cover, F_c. Due to the space limitations, we omit the proof.

3.2.2 Computing $p(r_i|R', F')$ based on the Cover

The following corollary provides us with a way to deal with inter-dependent features of a path when computing $p(r_i|R', F')$.

COROLLARY 3.1. *Let r_i be a result path, $F'(r_i)$ be the set of features of r_i, and $F'_c(r_i)$ be the corresponding feature cover. Then,*

$$\begin{aligned} p(r_i|R', F') &= p(F'(r_i)|R', F') = p(F'_c(r_i)|R', F') \\ &= \prod_{f \in F'_c(r_i)} p(f|R', F') = \prod_{f \in F'_c(r_i)} \frac{|R'(f)|}{|R'|}. \end{aligned}$$

In other words, the features in the cover are sufficient when computing the score of a path.

However, one more challenge remains in computing $p(r_i|R', F')$ based on the feature cover. While the above corollary is sufficient for computing $p(r_i|R', F')$ for paths that consist solely of features that occur in R', if any of the features does not occur in R', then we have $p(r_i|R', F') = 0$. This means that if any one of r_i's features does not occur in R', it is not distinguishable from other such paths, irrespective of how significant all its other features are. Naturally, this is not appropriate when ranking paths that contain a feature for which no user feedback is applicable. To prevent 0-ing out of $p(r_i|R', F')$ for such paths, a lower-bound, α, is used: i.e., if a feature f does not occur in R' or when $R' = \emptyset$, then $p(f|R', F') = \alpha$. Here, α is a positive real number, such that $\alpha \ll 1/|R'|$; i.e., smaller than the $p(f|R')$ value of any single feature. In the experiments we use $\alpha = 1/|R'|^2$.

3.3 Complexity

Let the number of results presented to the user for feedback be K and let the average length of the result path be l. Let the size of the feedback cover be $|F_c|$. Thus, computing feature scores takes $O(Kl|F_c|)$ time, where each result is scanned and the features are counted. Once the features are counted, all results obtained in the previous iteration have to be re-scored. If the total number of results is $|R|$, then this is done in $O(|R||F_c|)$ time.

4. EXPERIMENTS

We implemented the *AXP* framework and the underlying techniques using Java. The experiments were performed on a Xeon(TM) 2.9GHz processor workstation with 2.00G RAM.

Data and queries. The experiment results we report here are on the TreeBank data set [2]. This data set is a data-centric collection in that it contains the part-of-speech tags of Wall Street Journal articles, but not the textual content itself. Thus, it is a suitable set for feedback over data-centric XML. INEX data set [1] is not used because it is document-centric. We used a portion of the TreeBank XML tree, with $\sim 400K$ nodes and with a deep (the maximum depth is ~ 30 and the average depth is ~ 8) recursive structure. Thus, the data set also provides a suitable environment for testing the effect of feedback based exploration of path structured data.

We randomly generated initial queries of the form "$//l_1//l_2$". Each data point presented in the feedback performance figures in this section is obtained by averaging the corresponding performance metric for multiple queries on the same data set.

Feedback and the ground truth. In order to evaluate the relevance feedback process, we used both simulated (as in [4]) and human-provided feedback. For each query, one of its results is randomly chosen to represent its *ground truth*; i.e., the path that best

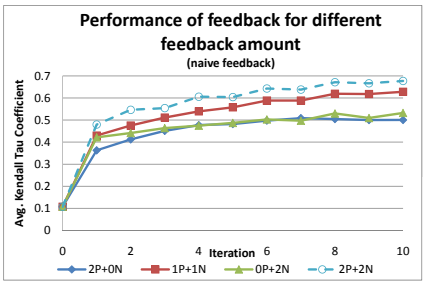

Figure 2: Effectiveness of feedback in matching ground truth; different curves denote different feedback amounts

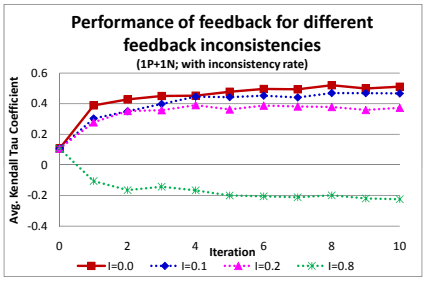

Figure 3: Effectiveness of feedback in matching ground truth; different curves denote different inconsistency rates

represents user's preferences. Given $K(= 20)$ result paths, those features that occur in the ground truth are candidates to be provided as positive feedback; features that do conflict with the ground truth are negative feedback candidates.

Efficiency and effectiveness measures. We report execution times as the measure of efficiency. The effectiveness of the feedback process can be evaluated by observing whether the rankings obtained through the feedback process are correlated with the ground truth: given a result path with a system assigned score, s, and with m mismatches with the ground truth (each missed or extra feature counts as a mismatch), the score should be *negatively correlated* with the number of mismatches. The standard measure of correlation, *Pearson correlation coefficient* assumes that the relationship between the two variables is linear. In *AXP*, we do not expect the result score and the number of mismatches will necessarily have a (inversely) linear relationship. Thus, we report *Kendall-tau rank coefficient* which does not make the assumption of linearity.

Evaluation of the feedback process. The results are collected using 10 different feedback sequences on 10 different queries.

(1) Effectiveness. The first set of feedback experiments shows the effectiveness of the positive and negative feedback in matching the rankings. As it can be seen in Figure 2, both positive and negative feedback improve the quality of the ranking. Negative feedback has a slight edge over positive feedback; this is because there are more negative features in the data that can cause mismatches with the ground truth than the positive features that can be missed. Nevertheless, using both positive and negative feedback together is the most effective approach for improving rankings (compare the `1P+1N` curve with the `2P+0N` and `0P+2N` curves). Providing more feedback (`2P+2N` curve) consistently improves the quality of the ranking, indicating that *AXP* is effective in getting the most out of the feedback statements.

We also ran a set of experiments to study the effects of potential inconsistencies in user feedback. For this purpose, a portion of the feedback provided to the system was modified to be inconsistent with the ground truth: in other words, the user is providing counter-productive feedback to this search goal. As shown in Figure 3, when the inconsistency of the user feedback increases, this negatively affects the quality of the ranking with respect to the base ground truth. In fact, when most of the feedback are inconsistent with the ground truth (see the "`I=0.8`" curve where the inconsistency rate in the feedback is 80%), the ranking coefficient becomes negative. This result is consistent with the expected behavior of feedback schemes and shows that our feedback process interprets the user's feedback correctly and can help the user when she is able to provide feedback that does not contain extreme inconsistency.

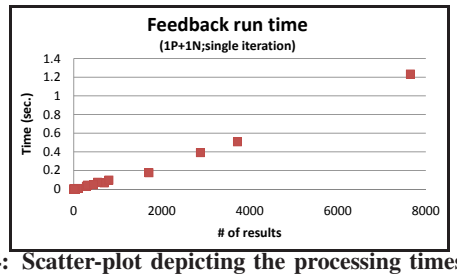

Figure 4: Scatter-plot depicting the processing times of feedback as a function of the number of results to be reranked

(2) Execution time. As it can be seen in Figure 4, the execution time of the feedback generation process is linear in the number of results to be reranked. The cost of the statistical analysis phase for the feedback is negligible.

(3) User study. Using the same data set and 7 users who are not knowledgeable about the data, we ran user studies. We provided these users 10 random queries and target results and asked them to try to reach the target result path using feedbacks. After single iteration, the user is presented top 10 result paths. With `1P+1N` and `2P+2N` feedback, the average Kendall-tau coefficients were 0.31 and 0.38 respectively. Without feedback the coefficient was 0.12. Feedback indeed helps improve the Kendall-tau coefficient of the results; more feedback helps achieve higher improvements.

5. CONCLUSIONS

This paper presented an *adaptive, and exploratory, path retrieval framework, AXP*, to help the user explore the path query results from data-centric XML collections, by providing positive and negative feedback that reflect her interests. Feedback are for ordering paths based on preference criteria. We developed a probabilistic path ranking scheme to deal with feedback statements. Experiments showed that feedback statements can be very effective in helping focus the exploration process.

6. REFERENCES

[1] Initiative for the evaluation of XML retrieval (INEX). http://www.inex.otago.ac.nz/.

[2] The penn treebank project, http://www.cis.upenn.edu/ treebank/.

[3] S. Amer-Yahia, D. Hiemstra, T. Roelleke, D. Srivastava, and G. Weikum. Db&ir integration: Report on the dagstuhl seminar "ranked xml querying". *SIGMOD Record*, 37(3):46–49, 2008.

[4] M. Ferecatu, M. Crucianu, and N. Boujemaa. Improving performance of interactive categorization of images using relevance feedback. In *ICIP (1)*, pages 1197–1200, 2005.

[5] L. Hlaoua, M. Boughanem, and K. Pinel-Sauvagnat. Combination of evidences in relevance feedback for xml retrieval. In *CIKM '07*.

[6] H. Pan, R. Schenkel, and G. Weikum. Fine-grained relevance feedback for xml retrieval. In *SIGIR '08*, pages 887–887, 2008.

[7] I. Ruthven and M. Lalmas. A survey on the use of relevance feedback for information access systems. *Knowl. Eng. Rev.*, 18(2), 2003.

[8] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.

[9] R. Schenkel and M. Theobald. Feedback-driven structural query expansion for ranked retrieval of xml data. In *EDBT*, 2006.

[10] R. Weber. Using relevance feedback in xml retrieval. In *Intelligent Search on XML Data*, pages 133–143, 2003.