XML Keyword Query Refinement

Jiaheng Lu‡ Zhifeng Bao†
‡School of Information and DEKE, MOE
Renmin University of China
jiahenglu,xfmeng@ruc.edu.cn

Tok Wang Ling† Xiaofeng Meng‡
†School of Computing
National University of Singapore
baozhife,lingtw@comp.nus.edu.sg

ABSTRACT

Existing works in XML keyword search have addressed the problem of finding matching results of a query. However, user input queries always contain irrelevant or mismatched terms, spelling errors etc, which causes the search results to be either empty or not meaningful. In this paper, we introduce the problem of XML keyword query refinement, and propose a set of effective and efficient solutions. Finally, extensive experiments show the efficiency and effectiveness of our approach.

Categories and Subject Descriptors

H.2.4 [**Information Systems**]: Database management— *Systems* ; H.3.3 [**Information Systems**]: Information storage and retrieval—*Information Search and Retrieval*

General Terms

Algorithms, Languages

Keywords

XML, keyword search, query refinement

1. INTRODUCTION

The success of keyword search engine in web has brought keyword search a very popular way for users to explore the information in database, as it allows users to pose free form queries[1, 3]. However, a user keyword query may often be an imperfect description of their real information need. Even when the information need is well described, a search engine may not be able to return the results matching the query as stated possibly due to term mismatch, keyword ambiguity, spelling error, etc. E.g. a user issues a query "database,proceeding" on XML data in Figure 1, but all results in XML data use a mismatched term "inproceedings" or "article", which causes an empty result to return.

The question then becomes whether we can offer a solution during search which automatically refines queries, in order to better represent users' search needs and help users more easily find the relevant information. This is what we mean by *XML keyword query refinement* as addressed in this paper.

To perform XML keyword query refinement *automatically*, there are two major technical challenges. The *first* challenge is that, it is unknown whether the refinement is required or not before processing the original query. A brute force approach [4, 5] needs submitting a query for an initial result retrieval, before deciding if the refinement operations should be used. However, it causes multiple scan of keyword inverted lists, which is inefficient; even worse, the suggested refined queries (generated by either user query log analysis or a consultation on a given refinement rule set) may not necessarily have any matching result in underlying database. The *second* challenge is, compared to IR keyword search whose search target is unexceptionally flat documents, XML keyword search needs to identify the most relevant components covering all query keywords, as the search target of an XML query is usually implicit and unfixed. Therefore, concrete search semantics and efficient methods to find matching results are an indispensable component of XML keyword query refinement.

In order to address the above two challenges, we first define four possible sources that cause ill-formed queries: (1) queries may contain misspelled or mismatched words (e.g. Q_1-Q_3,Q_5 in Table 1), (2) mistakenly split words (Q_6,Q_7), (3) mistakenly merged words. (4) queries that contain strong conjunctive constraints has no match against a small corpus (e.g. Q_4). These four ill-forms cause the results of the initial query either empty or irrelevant. As a result, four refinement operations, i.e. term substitution, term merging, term split and term deletion, are defined to increase the query coverage.

Second, we adopt a basic metric to judge the quality of a refined query RQ by computing the refinement cost from original query Q to RQ. As the RQ that has both minimal refinement cost and non-empty matching result over XML database is unknown ahead, it is prohibitively expensive to infer all RQ candidates and find the one with minimal refinement cost. Therefore, we design a dynamic programming solution to find the RQ with minimal refinement cost, and the potential Top-K RQ candidates are also produced as a side product.

Third, we propose a set of effective and efficient algorithms to find the optimal refined query with minimal refinement cost and its associated SLCA results.

Since our approaches are developed from a new perspective to combine query refinement and result generation, our algorithms are orthogonal to existing work on word stemming and segmentation [6], and can be used with existing ones in a complementary manner.

2. QUERY REFINEMENT

DEFINITION 2.1. *A keyword query Q issued on XML database D is said to* need refinement *if Q either has no matching result on*

KEYS'09, June 28, 2009, Providence, Rhode Island, USA.

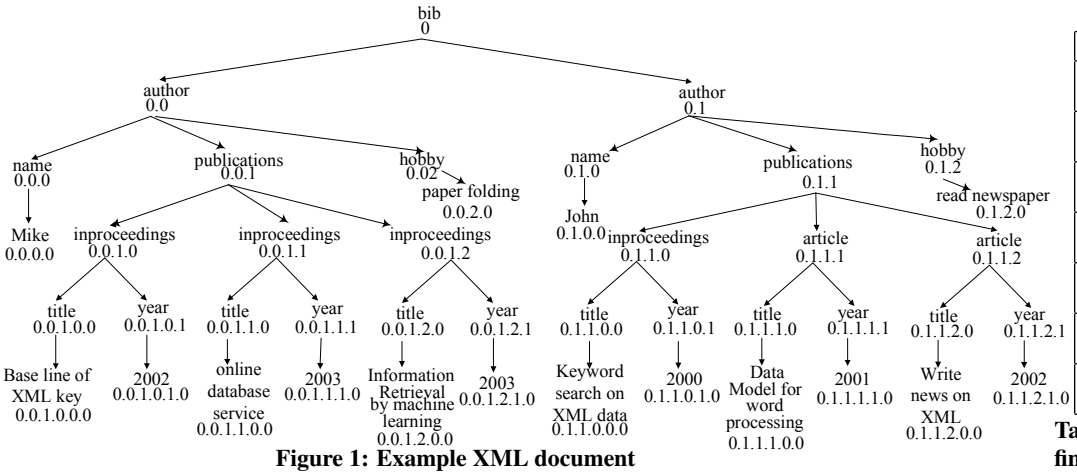

Figure 1: Example XML document

Original query	Refined query
Q_1: IR,2003,Mike	RQ_1: Information Retrieval,2003,Mike
Q_2: Mike,publication	RQ_2: Mike,publications
Q_3: Database,paper	RQ_3: Database, inproceedings
Q_4: XML,John,2003	RQ_4: XML,John
Q_5: machin,learn	RQ_5: machine, learning
Q_6: hobby,news,paper	RQ_6: hobby,newspaper
Q_7: on,line,data,base	RQ_7: online, database

Table 1: Query before and after refinement

D or all its matching results are the document root of D.

DEFINITION 2.2. *A refinement rule R associated with a refinement operation op is in form of:*

- $k_1,k_2,...,k_n \rightarrow k$, *if op = term merging*
- $k \rightarrow k_1,k_2,...,k_n$, *if op = term split*
- $k_1,k_2,...,k_n \rightarrow k'_1,k'_2,...,k'_m$, *if op = term substitution*

DEFINITION 2.3. *(**Top-K Query Refinement**) of an original query Q on XML database D returns the Top-K refined queries with their matching results in form of $RQList$={$<RQ_i,SLCA(RQ_i,D)>$| $i \in$[1,K]}, s.t. the following three properties hold:*

1. $\forall 1 \leq i < j \leq K$, *refineCost(Q,$RQ_i$)$\leq$refineCost(Q,$RQ_j$).*
2. $\forall i \in [1,K]$, $SLCA(RQ_i,D)$ *is not the root node of D.*
3. $\forall RQ' \notin RQList$, *refineCost(Q,$RQ'$) $\geq$ refineCost(Q,RQ_K).*

3. AUTOMATIC QUERY REFINEMENT AND RESULTS GENERATION

We propose three algorithms for XML keyword query refinement and results generation, that is, stack-based, partition-based and short-list-eager. We first propose a *stack-based approach* to find the optimal refined query with minimal refinement cost and its associated SLCA results. Based on the observation that the document root is never counted as a meaningful SLCA, we propose a *partition-based* approach, in which an XML document tree is divided into an ordered set of subtrees which we call as partitions, and query refinement is sequentially executed on each partition. Furthermore, we observe that the efficiency of the query refinement also depends on the distribution of keywords' frequency, and thus propose a *short-list-eager* approach to avoid the full scan of long keyword lists as much as possible, which works well when the frequency distribution of query keywords is skewed.

4. RANKING OF REFINED QUERIES

In Sec. 3, each algorithm judges the quality of a refined query RQ w.r.t original query Q according to the *refinement cost* of reformulating Q to RQ, which is a preliminary step towards a relevance-oriented ranking model. However, if multiple RQ candidates have the same refinement cost, how to rank them is an interesting problem unaddressed so far. E.g. Q={XML, Jim, 2001} can be refined as {XML, 2001}, {Jim, 2001} or {XML, Jim} via one term deletion. Thus, it is necessary to design an elaborate query ranking model.

Guideline 1: The more matching results a refined query RQ has on D, the higher the rank of RQ.

Guideline 2:The more discriminative a keyword k_i is w.r.t the original query Q on D, the lower the rank of RQ without k_i.

Guideline 3: For an original query Q that has multiple desired "search for" node type T ([1]), the higher the confidence $C_{for}(T,Q)$ as a desired "search for" node is, the higher the rank of $\rho(RQ,Q,T)$.

Guideline 4: The less refinement cost needed to transform Q to RQ, the higher the rank of RQ.

We have designed a query ranking model to incorporate all the above guidelines, by which an overall ranking formula is derived.

5. EXPERIMENTS

We have three observations in our experiments. (1) Partition outperforms both stack-based and short-list-eager for almost all queries by 60% and 50% resp. (2) Stack-based is the least efficient one, as it has to compute refinement cost for each stack entry. (3) Through the Cumulated Gain-based evaluation (CG) [2], we show that the ranking model can effectively finding the top-k answers. In particular, Guideline 1 plays a more important role than Guideline 2 and 3 to find top-1 answer and all four guidelines are almost equivalently important for top-4 answers.

6. ACKNOWLEDGEMENTS

This research was partially supported by National 863 High-Tech Research Plan (No: 2009AA01Z133, 2007AA01Z155, 2009AA01Z149), and Key project in Ministry of Education (MOE), China (109004) and NSF China (60833005, 60573091).

7. REFERENCES

[1] Z. Bao, T. W. Ling, B. Chen, and J. Lu. Effective xml keyword search with relevance oriented ranking. In *ICDE*, 2009.

[2] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*

[3] Z. Liu and Y. Chen. Identifying meaningful return information for xml keyword search. In *SIGMOD*, 2007.

[4] Y. Mass and M. Mandelbrod. Component ranking and automatic query refinement for xml retrieval. In *Proceedings of the 3rd INEX Workshop*, pages 73–84, 2004.

[5] H. Pan, A. Theobald, and R. Schenkel. Query refinement by relevance feedback in an xml retrieval system. In *ER*, pages 854–855, 2004.

[6] K. Q. Pu and X. Yu. Keyword query cleaning. *PVLDB*, 1(1):909–920, 2008.