FUZZY KEYWORD SEARCH IN XML DATA

Ranjeeth Kumar.M[1]

Dept of CSE, Vaagdevi college of engineering
Bollikunta, Warangal, AP, India.
maduri.ranjith@gmail.com

M.Suresh Kumar[2]

Dept of CSE, Vaagdevi college of engineering
Bollikunta, Warangal, AP, India.
mandala.suresh83@gmail.com

S.S.V.N.Sarma[3]

Dept of IT, Vaagdevi college of engineering
Bollikunta, Warangal, AP, India.
Ssvn.sarma@gmail.com

Abstract - In a traditional keyword search system over XML data, a user make a keyword query, submits it to the system, and retrieves relevant answers. In this case where the user has limited knowledge about the data, often the user feels left in the dark when issuing queries, and has to use a try and see approach for finding information. We study fuzzy keyword search in XML data, a new information access prototype in which the system searches XML data on the wing as the user types in query keywords. It allows users to explore data as they type, even in the presence of minor typos and format inconsistencies of their keywords. In this paper, for the first time we formalize and solve the problem of effective fuzzy keyword search over XML data. It extends Auto complete by supporting queries with multiple keywords in XML data. It can find high superiority answers that have keywords matching query keywords approximately. Our effective index structures and searching algorithms can achieve a very high interactive speed. We study research challenges in this new search construction.

Keywords: fuzzy, fuzzy logic, fuzzy keyword sets, fuzzy keyword search.

I. Introduction

Traditional methods use query languages such as XPath and XQuery to query XML data. These methods are powerful but unfriendly to the user with limited knowledge about the data. First, these query languages are hard to realize to the user with limited knowledge about the database. For example, XQuery is somewhat complicated to grasp. Second, these languages require the queries to be posed against the underlying, sometimes complex, database schemas. Fortunately, keyword search is proposed as an alternative means for querying XML data, which is simple and yet familiar to most Internet users as it only requires the input of keywords. Keyword search is a widely accepted search paradigm for querying document systems and the World Wide Web. Recently, the database research community has been studying challenges related to keyword search in XML data. One important advantage of keyword search is that it enables to search information without knowing a complex query language such as XPath or XQuery, or having aforementioned knowledge about the structure of the underlying data to the needs of that user. Thus the drawbacks of existing schemes signifies the important need for new techniques that support searching flexibility, tolerating both minor typos and format inconsistencies. Fuzzy keyword search greatly enhances system usability by returning the matching files when user is searching inputs exactly match the predefined keywords or the closest possible matching files based on keyword similarity semantics, when *exact* match fails. More specifically, we use edit distance to quantify keywords similarity and develop a novel technique, i.e., a wildcard-based technique, for the construction of fuzzy keyword sets. This technique eliminates the need for enumerating all the fuzzy keywords and the resulted size of the fuzzy keyword sets is significantly reduced. Based on the constructed fuzzy keyword sets, we propose an efficient fuzzy keyword search scheme. Fuzzy keyword sets, we propose an efficient fuzzy keyword search scheme. It is quite common that users' searching input might not exactly match those pre-set keywords due to the possible typos, such as Illinois and Ilinois, representation inconsistencies, such as PO BOX and P.O. Box, and/or her lack of exact knowledge about the data. The naive way to support fuzzy keyword search is through simple spell check mechanisms. This approach does not completely solve the problem and sometimes can be ineffective due to the following reasons on the one hand it requires additional interaction of user to determine the correct word from the candidates generated by the spell check algorithm, which unnecessarily costs user's extra computation effort on the other hand, in case that user accidentally types some other valid keywords by mistake for example, search for "hat" by carelessly typing "cat" the spell check algorithm would not even work at all, as it can never differentiate between two actual valid words. Thus, the drawback of existing schemes signifies the important need for new techniques that support searching flexibility, tolerating both minor typos and format inconsistencies. We show that the proposed solution is secure and privacy preserving, while correctly realizing the goal of fuzzy keyword search.

I.I. Motivation

The main goal of this project is to focus on fuzzy keyword search in XML data. To the best of our knowledge, we formalize for the first time the problem of effective fuzzy keyword search over XML data while maintaining keyword privacy.

I.II. Problem Definition

A. System Model

In this paper, we consider a collection of n XML data files $C = (F_1, F_2, \ldots, F_N)$ stored in the server, a predefined set of distinct keywords $W = \{w_1, w_2, \ldots, w_p\}$, the server provides the search service for the authorized users over the XML data C. The server is responsible for mapping the searching request to a set of data files, where each file is indexed by a file ID and linked to a set of keywords. The fuzzy keyword search scheme returns the search results according to the following rules: 1) if the user's searching input exactly matches the pre-set keyword, the server is expected to return the files containing the keyword1; 2) if there exist typos and/or format inconsistencies in the searching input, the server will return the closest possible results based on pre-specified similarity semantics (to be formally defined in section I-II-D). Note that we do not differentiate between files and file IDs in this paper.

B. Threat Model

We consider a semi-trusted server. When designing fuzzy keyword search scheme, we will follow the definition deployed in the traditional search [8]. More specifically, it is required that nothing should be leaked from the remotely stored files and index beyond the outcome and the pattern of search queries.

C. Design Goals

In this paper, we address the problem of supporting efficient yet privacy-preserving fuzzy keyword search services over XML data. Specifically, we have the following goals: i) To explore new mechanism for constructing storage efficient fuzzy keyword sets; ii) To design efficient and effective fuzzy search scheme based on the constructed fuzzy keyword sets.

D. Preliminaries

Edit Distance is a method of quantitatively measure the string similarity. In this paper, we resort to the well-studied edit distance [11] for our purpose. The edit distance ed(w_1, w_2) between two words w_1 and w_2 is the number of operations required to transform one of them into the other. The three primitive operations are 1) Substitution: changing one character to another in a word; 2) Deletion: deleting one character from a word; 3) Insertion: inserting a single character into a word. Given a keyword w, we let S_w,d denote the set of words w' satisfying ed(w,w') $\leq d$ for a certain integer d. Fuzzy Keyword Search Using edit distance, the definition of fuzzy keyword search can be formulated as follows: Given a collection of n XML data files $C = (F_1, F_2, \ldots, F_N)$ stored in the server, a set of distinct keywords $W = \{w_1, w_2, \ldots, w_p\}$ with predefined edit distance d, and a searching input (w, k) with edit distance k ($k \leq d$), the execution of fuzzy keyword search returns a set of file IDs whose corresponding data files possibly contain the word w, denoted as $FIDw$: if $w = w_i \in W$, then return $FIDw_i$; otherwise, if $w' \in W$, then return $\{FIDw_i\}$, where ed(w,w_i) $\leq k$. Note that the above definition is based on the assumption that $k \leq d$. In fact, d can be different for distinct keywords and the system will return $\{FIDw_i\}$ satisfying ed(w,w_i) $\leq$ min$\{k, d\}$ if exact match fails.

II. The Straightforward Approach

Before introducing our construction of fuzzy keyword sets, we first propose a straightforward approach that achieves all the functions of fuzzy keyword search, which aims at providing an overview of how fuzzy search scheme works over XML data. The scheme of the fuzzy keyword search goes as follows: We begin by constructing the fuzzy keyword set S_{wi},d for each keyword $w_i \in W$ ($1 \leq i \leq p$) with edit distance d. The intuitive way to construct the fuzzy keyword set of w_i is to enumerate all possible words w'_i that satisfy the similarity criteria ed(w_i, w'_i) $\leq d$, that is, all the words with edit distance d from w_i are listed. For example, the following is the listing variants after a substitution operation on the first character of keyword FUZZY: {AUZZY, BUZZY, CUZZY, · · · , YUZZY, ZUZZY}. Based on the resulted fuzzy keyword sets, the fuzzy search over XML data is conducted as follows: 1) To build an index for w_i,; 2) Upon receiving the search request S_w, the server compares it with the index table and returns all the possible file identifiers according to the fuzzy keyword definition. The user returned results and retrieves relevant files of interest. This straightforward approach apparently provides fuzzy keyword search over the XML files while achieving search privacy. However, this approach has serious efficiency disadvantages. The simple enumeration method in constructing fuzzy keyword sets would introduce large storage complexities, which greatly affect the usability. Recall that in the definition of edit distance, substitution, deletion and insertion are three kinds of operations in computation of edit distance. The numbers of all similar words of w_i satisfying ed(w_i, w'_i) $\leq d$ for d = 1, 2 and 3 are approximately $2k \times 26$, $2k^2 \times 26^2$, and $4/3k^3 \times 26^3$, respectively. For example, assume there are 104 keywords in the file collection with average keyword length 10, d = 2, and the output length of hash function is 160 bits, then, the resulted storage cost for the index will be 30GB. Therefore, it brings forth the demand for fuzzy keyword sets with smaller size.

III. Literature Survey

III.I. Wildcard – Based Technique

In wildcard-based technique, all the variants of the keywords have to be listed even if an operation is performed at the same position. Based on the above observation, we proposed to use a wildcard to denote edit operations at the same position. The wildcard based fuzzy set edit's distance to solve the problems.

III.II. Gram-Based Technique

Another efficient technique for constructing fuzzy set is based on grams. The gram of a string is a substring that can be used as a signature for efficient approximate search. While gram has been widely used for constructing inverted list for approximate string search, we use gram for the matching purpose.

IV. Existing System and Proposed System

IV.I. Existing System

This straightforward approach apparently provides fuzzy keyword search over the XML data. However, this approaches serious efficiency disadvantages. The simple enumeration method in constructing fuzzy keyword sets would introduce large storage complexities, which greatly affect the usability.

IV.II. Proposed System

Main Modules:

- Wildcard – Based Technique
- Gram - Based Technique
- Construction of Effective Fuzzy Keyword Search in XML data

IV.II.I. Wildcard – Based Technique

Based on the above observation, we proposed to use a wildcard to denote edit operations at the same position. The wildcard-based fuzzy set edits distance to solve the problems. For example, for the keyword CASTLE with the preset edit distance 1, its wildcard based fuzzy keyword set can be constructed as **FUZZY, 1 = {FUZZY, *FUZZY, *UZZY ………. FUZZ*, FUZZY*}.**

IV.II.II. Gram – Based Technique

Another efficient technique for constructing fuzzy set is based on grams. The gram of a string is a substring that can be used as a signature for efficient approximate search.

While gram has been widely used for constructing inverted list for approximate string search, we use gram for the matching purpose. We propose to utilize the fact that any primitive edit operation will affect at most one specific character of the keyword, leaving all the remaining characters untouched. In other words, the relative order of the remaining characters after the primitive operations is always kept the same as it is before the operations. For example, the gram-based fuzzy set FUZZY, 1 for keyword FUZZY can be constructed as **{FUZZY, FUZZ, FUZY, FUZZ, UZZY, FUZ, FUY ...Y}.**
Edit Distance: Substitution**,** Deletion**,** and Insertion.

IV.II.III. Construction of Effective Fuzzy Keyword Search in XML data

The key idea behind our secure fuzzy keyword search is two-fold: 1) Building up fuzzy keyword sets that incorporate not only the exact keywords but also the ones differing slightly due to minor typos, format inconsistencies, etc. 2) Designing an efficient and secure searching approach for keyword retrieval based on the resulted fuzzy keyword sets.

A. Advanced Technique for Constructing Fuzzy Keyword Sets

To provide more practical and effective fuzzy keyword search constructions with regard to both storage and search efficiency, we now propose an advanced technique to improve the straightforward approach for constructing the fuzzy keyword set. Without loss of generality, we will focus on the case of edit distance d = 1 to elaborate the proposed advanced technique. For larger values of d, the reasoning is similar. Note that the technique is carefully designed in such a way that while suppressing the fuzzy keyword set, it will not affect the search correctness.

Wildcard-based Fuzzy Set Construction In the above straightforward approach, all the variants of the keywords have to be listed even if an operation is performed at the same position. Based on the above observation, we proposed to use a wildcard to denote edit operations at the same position. The wildcard-based fuzzy set of *wi* with edit distance d is denoted as $S_{wi,d}=\{S_{wi,0}, S_{wi,1}, \cdots, S_{wi,d}\}$, where $S_{wi,\tau}$ denotes the set of words w_i with τ wildcards. Note each wildcard represents an edit operation on w_i. For example, for the keyword FUZZY with the pre-set edit distance 1, its wildcard-based fuzzy keyword set can be constructed as FUZZY, 1 = {FUZZY, *FUZZY, *UZZY, F*UZZY, F*ZZY, · · · , FUZZ*Y, FUZZ*, FUZZY*}. The total number of variants on FUZZY constructed in this way is only 12+1, instead of 12 × 26 + 1 as in the above exhaustive enumeration approach when the edit distance is set to be 1. Generally, for a given keyword w_i with length size of S_{wi},1 will be only 2l + 1 + 1, as compared to (2l + 1) × 26 + 1 obtained in the straightforward approach. The larger the pre-set edit distance, the more storage overhead can be reduced with the same setting of the example in the straightforward approach; the proposed technique can help reduce the storage of the index from 30GB to approximately 40MB. In case the edit distance is set to be 2 and 3, the size of S_{wi},2 and S_{wi},3 will be $C^1_l+1+C^1_l\cdot C^1_l+2C^2_l+2$ and $C^1_l + C^3_l + 2C^2_l + 2C^2_l\cdot C^1_l$. In other words, the number is only $O(l^d)$ for the keyword with length l and edit distance d.

B. The Efficient Fuzzy Keyword Search Scheme

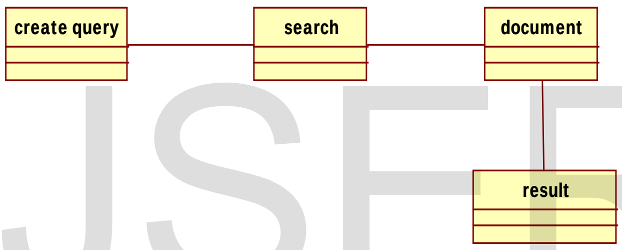
Based on the storage-efficient fuzzy keyword sets, we show how to construct an efficient and effective fuzzy keyword search scheme. The scheme of the fuzzy keyword search goes as follows:

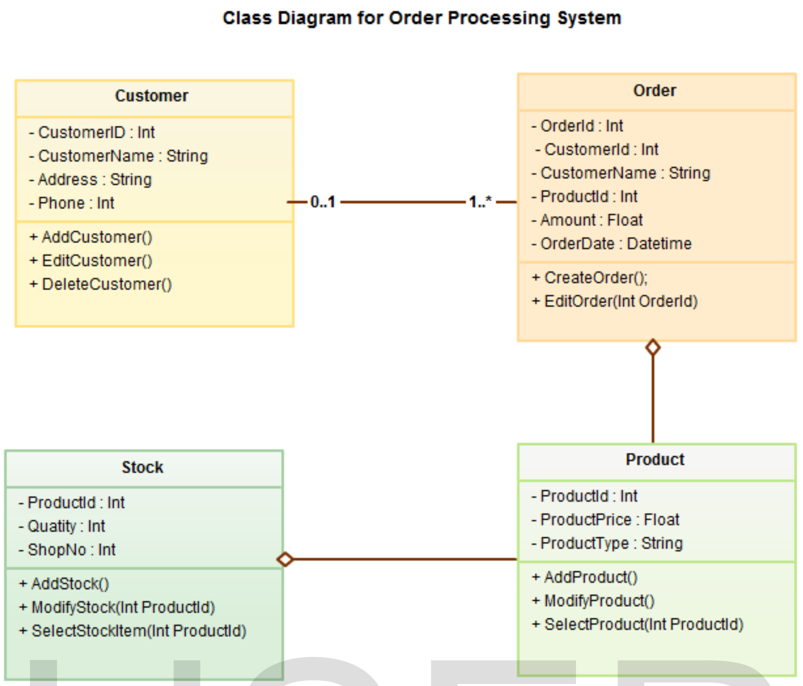
1) To build an index for *wi* with edit distance *d*, the data owner first constructs a fuzzy keyword set S_{wi},*d* using the wildcard based technique and gram based technique then data files are outsourced for storage.

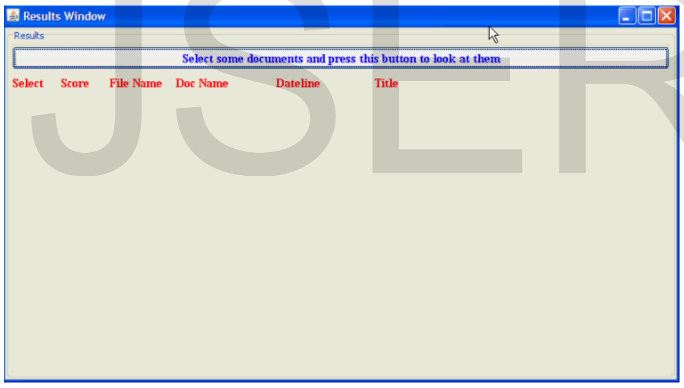
2) To search with (*w*, *k*), the authorized user computes the set $\{S_{w'}\}_{w \in Sw,k}$, where $S_{w,k}$ is also derived from the wildcard-based and gram based fuzzy set construction.

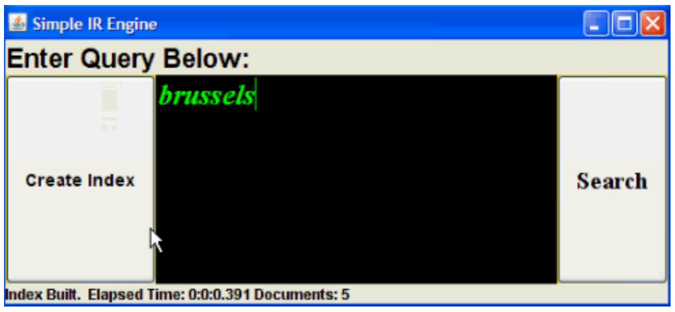
3) Upon receiving the search request $\{S_{w'}\}_{w \in Sw,k}$, the server compares them with the index table and returns all the possible file identifiers according to the fuzzy keyword definition in **I.II.D** section. The retrieves relevant XML files of interest. In this construction, the technique of constructing search request for *w* is the same as the construction of index for a keyword. As a result, the search request is a set based on $S_{w,k}$, instead of a single word as in the straightforward approach. In this way, the searching result correctness can be ensured.

V. System Design

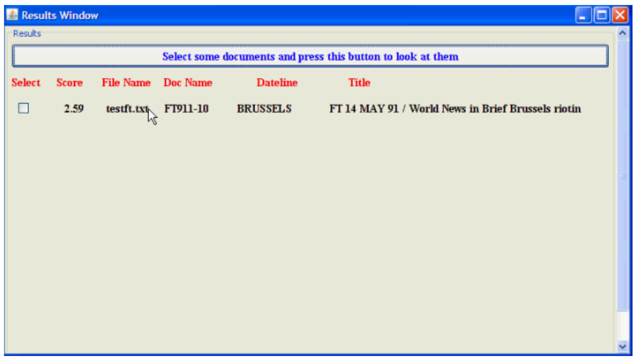




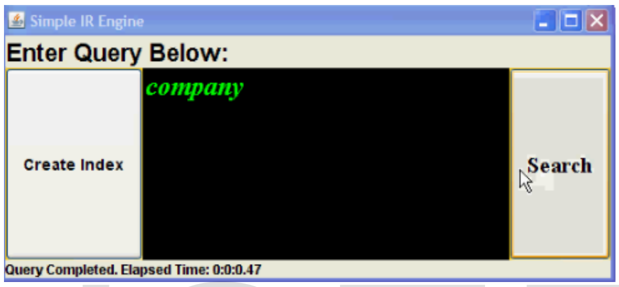




Index Built. Elapsed Time: 0:0:0.391 Documents: 5

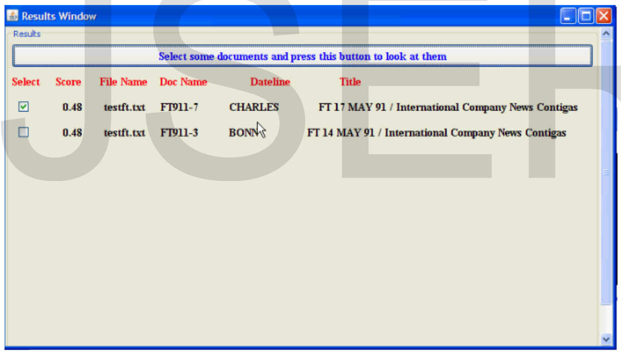


Select	Score	File Name	Doc Name	Dateline	Title
☐	2.59	testft.txt	FT911-10	BRUSSELS	FT 14 MAY 91 / World News in Brief Brussels riotin

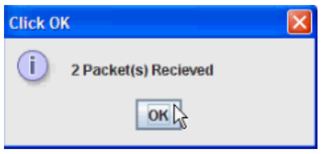


company

Query Completed. Elapsed Time: 0:0:0.47



Select	Score	File Name	Doc Name	Dateline	Title
✓	0.48	testft.txt	FT911-7	CHARLES	FT 17 MAY 91 / International Company News Contigas
☐	0.48	testft.txt	FT911-3	BONN	FT 14 MAY 91 / International Company News Contigas



2 Packet(s) Recieved

VI. Conclusion and Future Work

In this project, for the first time we formalize and solve the problem of fuzzy keyword search for achieving effective utilization of remotely stored XML data. We examined the LCA-based method to interactively identify the predicted answers. We have developed a minimal-cost-tree-based search method to efficiently and progressively identify the most relevant answers. We proposed a heap-based method to avoid constructing union lists on the fly. We devised a forward-index structure to further improve search performance. We have implemented our method, and the experimental results show that our method achieves high search efficiency and result quality We design an advanced technique (i.e., wild card-based technique and Gram based technique) to construct the storage efficient fuzzy keyword sets by exploiting a significant observation on the similarity metric of edit distance. Based on the constructed fuzzy keyword sets, we further propose an efficient fuzzy keyword search scheme. Through rigorous security analysis, Future work is on security mechanisms that support search semantics that takes into consideration conjunction of keywords, sequence of keywords, and even the complex natural language semantics to produce highly relevant search results and search ranking that sorts the searching results according to the relevance criteria.

VII. References

1. A. Balmin, V. Hristidis, and Y. Papakonstantinou, “Objectrank: Authority-Based Keyword Search in Databases,” Proc. Int’l Conf. Very Large Data Bases (VLDB), pp. 564-575, 2004.
2. C. Li, J. Lu, and Y. Lu, “Efficient merging and filtering algorithms for approximate string searches,” in Proc. of ICDE’08, 2008.
3. D. Boneh and B. Waters, “Conjunctive, subset, and range queries,” in Proc. of TCC’07, 2007, pp. 535–554.
4. D. Song, D. Wagner, and A. Perrig, “Practical techniques for searches,” in Proc. of IEEE Symposium on Security and Privacy’00, 2000.
5. E.-J. Goh, “Secure indexes,” Cryptology ePrint Archive, Report 003/216, 2003, http://eprint.iacr.org/.
6. Google, “Britney spears spelling correction,” Referenced online at http://www.google.com/jobs/britney.html,June 2009.
7. H. Bast and I. Weber, “Type Less, Find More: Fast Autocompletion Search with a Succinct Index,” Proc. Ann. Int’l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR), pp. 364-371, 2006.
8. H. Bast and I. Weber, “The Completesearch Engine: Interactive, Efficient, and towards Ir&db Integration,” Proc. Biennial Conf. Innovative Data Systems Research (CIDR), pp. 88-95, 2007.
9. J. Feigenbaum, Y. Ishai, T. Malkin, K. Nissim, M. Strauss, and R. N.Wright “Secure multiparty computation of approximations,” in Proc. of ICALP'01.
10. M.D. Atkinson, J.-R. Sack, N. Santoro, and T. Strothotte, “Min-max Heaps and Generalized Priority Queues,” Comm. ACM, vol. 29, no. 10, pp. 996-1000, 1986.
11. R. Ostrovsky, “Software protection and simulations on oblivious rams,” Ph.D dissertation, Massachusetts Institute of Technology, 1992.
12. S. Agrawal, S. Chaudhuri, and G. Das, “Dbxplorer: A System for Keyword-Based Search over Relational Databases,” Proc. Int’l Conf. Data Eng. (ICDE), pp. 5-16, 2002.
13. 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, vol. 37, no. 3, pp. 46- 49, 2008.
14. V. Levenshtein, “Binary codes capable of correcting spurious insertions and deletions of ones,” Problems of Information Transmission, vol. 1, no.1, pp. 8–17, 1965.
15. Y.-C. Chang and M. Mitzenmacher, “Privacy preserving keyword searches on remote stored data,” in Proc. of ACNS’05, 2005.
16. Z. Bao, T.W. Ling, B. Chen, and J. Lu, “Effective XML Keyword Search with Relevance Oriented Ranking,” Proc. Int’l Conf. Data Eng. (ICDE), 2009.