Information Systems 47 (2015) 178–193

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

Top-k-size keyword search on tree structured data

Aggeliki Dimitriou [a], Dimitri Theodoratos [b,*], Timos Sellis [c]

[a] School of Electrical and Computer Engineering National Technical University of Athens, Greece
[b] Department of Computer Science New Jersey Institute of Technology, United States
[c] School of Computer Science and Information Technology, RMIT University, Australia

ARTICLE INFO

Article history:
Received 22 July 2013
Received in revised form
14 March 2014
Accepted 14 July 2014
Recommended by: Xifeng Yan
Available online 25 July 2014

Keywords:
Keyword search
Tree-structured data
LCA
Search algorithm
Ranking

ABSTRACT

Keyword search is the most popular technique for querying large tree-structured datasets, often of unknown structure, in the web. Recent keyword search approaches return lowest common ancestors (LCAs) of the keyword matches ranked with respect to their relevance to the keyword query. A major challenge of a ranking approach is the efficiency of its algorithms as the number of keywords and the size and complexity of the data increase. To face this challenge most of the known approaches restrict their ranking to a subset of the LCAs (e.g., SLCAs, ELCAs), missing relevant results.

In this work, we design novel top-k-size stack-based algorithms on tree-structured data. Our algorithms implement ranking semantics for keyword queries which is based on the concept of LCA size. Similar to metric selection in information retrieval, LCA size reflects the proximity of keyword matches in the data tree. This semantics does not rank a predefined subset of LCAs and through a layered presentation of results, it demonstrates improved effectiveness compared to previous relevant approaches. To address performance challenges our algorithms exploit a lattice of the partitions of the keyword set, which empowers a linear time performance. This result is obtained without the support of auxiliary precomputed data structures. An extensive experimental study on various and large datasets confirms the theoretical analysis. The results show that, in contrast to other approaches, our algorithms scale smoothly when the size of the dataset and the number of keywords increase.

1. Introduction

Tree-based structures (e.g., XML, JSON, YAML) are a widely adopted format for exporting and exchanging data on the web. Keyword search is the most popular technique for retrieving information from the web because it frees the users from (a) mastering a complex query language (e.g., XQuery), and (b) having full knowledge of the schemas of the data sources they want to query. In contrast to keyword search on flat text documents, keyword search on XML (or other tree-structured) data returns not whole documents but appropriately selected fragments of XML trees that contain matches to all the keywords [28,14]. A large number of publications elaborate on the form [14,24,4,5,8,26,23] and the meaningful instances of these result fragments [14,10,22,31,15,29,18,32,25,26,30,17] in the input XML tree. Usually, the query results are the minimum connecting trees that contain one instance of every keyword in the query. These minimum connecting trees are represented by their root which is the lowest common ancestor (LCA) of the included keyword instances. Approaches that select and return as answer to keyword queries a subset of the LCAs in the XML tree are called *filtering* because they filter out LCAs that are considered irrelevant [10,22,31,18,32]. Although, filtering approaches are intuitively reasonable, they are sufficiently ad hoc and they are frequently violated in practice resulting in low precision and/or recall [30].

* Corresponding author.
E-mail addresses: angela@dblab.ntua.gr (A. Dimitriou), dth@cs.njit.edu (D. Theodoratos), timos.sellis@rmit.edu.au (T. Sellis).

http://dx.doi.org/10.1016/j.is.2014.07.002
0306-4379/

A better approach would *rank* the LCAs placing on top those that are considered more relevant to the query. Ranking the LCAs greatly improves the usability of the system. Most ranking approaches are based on strategies employed for flat text documents (e.g., *tf*∗*idf* or PageRank) adapted to the hierarchical nature of the XML trees [14,10,30,2]. Recognizing the fact that users are usually interested in a small number of query results, some papers recently develop *top-k* algorithms for keyword search over XML data [19,8,21]. The goal of the top-k algorithms is to rank and select the top-k results without explicitly producing and ranking all of them.

Current approaches for keyword search on tree-structured data face a number of problems:

Problem 1: *Performance scalability*. The number of LCAs for a given keyword query can be very large. Even though multiple query matches can share the same LCA, this number can, in the worst case, be exponential on the size of the query (number of keywords). The complexity of previous algorithms that process and possibly rank the totality of the LCAs depends on the product of the size of the keyword inverted lists [15,30,23]. Consequently, such algorithms do not scale satisfactorily when the size of the dataset and the number of keywords increase.

Problem 2: *Dependence from additional auxiliary data structures*. In order to address the performance scalability problem a number of approaches rely on the construction of auxiliary data structures, on top of the keyword inverted lists (e.g., B+-tree [32], ranked Dewey inverted list and B+-tree [14], data summary index [23], hash count index [33]). Moreover, many approaches rely on additional auxiliary data structures and statistical information even for efficiently implementing query semantics (e.g., interconnection index [10,9], normalized total correlation [30]). Building these auxiliary structures requires producing and storing the inverted lists and the auxiliary data structures of the dataset. This process is not only time consuming but also renders these approaches impractical to a number of applications including streaming applications.

Problem 3: *Quality of the answer*. In order to avoid producing a large number of LCAs the different ranking and top-k approaches proposed [14,10,8,21] produce and rank not all the LCAs but only a small subset defined by filtering semantics (e.g., SLCA [31,15,29,25], ELCA [14,32,33]). This strategy, even though computationally appealing, is semantically insufficient since, despite the potential quality of the ranking criteria, it penalizes the query answer with the deficiencies of the corresponding filtering semantics. For instance, if relevant results are missed by the filtering semantics they cannot be recovered and presented to the user, no matter how good or efficient the ranking technique is.

Problem 4: *User interface of top-k approaches*. In order to support the users in coping with a possibly large number of results but also for performance reasons, recent approaches return only top-k results to keyword queries [8,21]. However, the selection of k by the user at query time is a tricky issue: a selection of a small k may miss relevant results while a selection of a large k may overwhelm the user with a large number of irrelevant results and unnecessarily increase the response time. Using a good ranking function could address the problem of the multitude of returned results but it does not resolve the performance issue. Successfully selecting the appropriate k requires knowledge of the number of results, which depends on the size and structure of the dataset, the number of query keywords and the keyword frequencies in the dataset. Requiring the user to have detailed statistical information about the dataset and be able to apply complex techniques for estimating the number of relevant results defies the reason for using such a simple query language as keyword queries.

Our approach. In this paper, we present novel, efficient top-k algorithms which compute results of top-k LCA sizes. This approach contrasts to traditional top-k approaches which focus on computing top-k results. The concept of LCA size introduced in this paper reflects the proximity of keywords in tree-structured data and, similar to the concept of keyword proximity in the IR domain, is used here as a relevance criterion for the results of a keyword query.

The efficiency of our algorithms is achieved by exploiting a lattice of keyword partitions of the query keywords. The paths of the lattice are used in gradually combining keyword instances into partial LCAs, allowing the exclusion of combinations of other instances of the same keywords with larger size before they contribute to the formation of full LCAs. This technique avoids the exhaustive computation of all keyword instance combinations in finding the LCAs of top-k sizes. As a consequence, our algorithms scale smoothly when the size of the dataset increases and tackle successfully the performance scalability issue (Problem 1). Interestingly, our algorithms achieve efficiency without recurring to the construction of auxiliary data structures, this way avoiding the preprocessing phase required by other approaches (Problem 2).

Our algorithms implement ranking semantics and consider for ranking the full set of LCAs. Therefore, they do not suffer from deficiencies (low precision or recall) of previous ranking and top-k approaches which are restricted to a predefined, structurally determined subset of LCAs (Problem 3). The returned results are grouped in LCA size layers which are ranked on LCA size. The layers can have a varying number of results. This layered computation relieves the user from providing with the query an appropriate k for top-k result computation (Problem 4). The top-1-size layer contains results of highest proximity and shows high precision while recurrence to a subsequent layer is possibly needed only in case increased recall is desired.

Contribution. The main contributions of our paper are the following:

- We design novel, efficient multi-stack based algorithms, that exploit a lattice of stacks representing the different partitions of query keywords. Our algorithms compute: (a) keyword search results below a given LCA size threshold, (b) results of top-k LCA sizes and (c) top-k results.
  In contrast to previous approaches, ours does not involve auxiliary index structures and therefore it can be exploited on datasets which have not been preprocessed.
- We analyze our algorithms and show that for a fixed number of query keywords their performance is linear on the size of the input keyword inverted lists. This behavior contrasts with that of previous algorithms,