ABSTRACT

For keyword search on XML data, traditionally, a list of query results in the form of subtrees will be returned to users. However, we find that it is still not sufficient to meet users’ information needs because: (1) the search intention of a certain keyword query varies from person to person; (2) amongst the query results, they may have sibling or containment relationships (in the context of whole XML database), which could be important for users to digest the query results and should be shown to users. Therefore, we try to equip the traditional XML keyword search engine with our new exploration model XMAP, providing user an interactive yet novel way to explore the results with better user experience.

Categories and Subject Descriptors
H.5.2 [User Interfaces]: Graphical user interfaces (GUI)

Keywords
XML keyword search; query results

1. INTRODUCTION

Structured/semi-structured data, such as XML data, are usually stored in database which are difficult for user to manipulate, and it is demanding to improve the usability of such data [3]. Keyword search enables a way to access the data, but it is far away from user’s expectation of being able to explore and manipulate the data directly. For instance, current research efforts on XML keyword search mainly focus on defining what should be returned as results. E.g., given the XML data in Figure 1, the left hand side of Figure 3 shows what will be returned by existing keyword search methods. As XML data is usually modeled as a rooted tree, the very basic search semantics is to find the subtree whose root node is the lowest common ancestor (LCA) of all query keywords, upon which various search semantics such as Smallest LCA (SLCA) and Exclusive LCA (ELCA) are proposed.

Visual query interface has been proved as one effective way to improve the data usability [3]. We are motivated to integrate the power of graphical query interface and keyword search. As a result, we present XMAP, an interactive exploration model for user to visualize, manipulate and further explore the query results.

Motivation 1 For a certain keyword query, its search intention can be various and differs from user to user issuing it. It is very difficult for a single search semantics to capture all possible search intentions for a certain keyword query.

EXAMPLE 1. An XML tree in Figure 1 describes the chain-store information of a company. To facilitate our discussion, each node in the figure is assigned a unique ID. For a query “Allen female”, whose search intention is somehow ambiguous, one of the results by LCA is the leftmost subtree rooted at node r3.Store. It meets the information needs for those who want to find the information of a cashier. But for people who may want to (further) find the chain-store with cashier Allen, they need some “ways” to make adjustment to the query results in order to find what they want.

So far, the only way for a user is to revise and re-submit the keyword query, which indeed is not trivial. For the case in Example 1, the user intending to search for chain-stores need to add more keywords to make her search intention more specific. Since the user may not know much about the database, she may not know what keywords to add in. There are a lot of choices: “store”, “shop”, “chain-store”, etc. From Figure 1 we can find, the information of a chain-store (which user wants) is contained in the subtree just above the current result (i.e. the subtree rooted at r3). To reach what she wants, the user only needs to navigate upward from the current query result.

Motivation 2 All the data in an XML tree are inter-connected by the hierarchical structure. Thus, each query result is a part of the XML tree, and many of them may have sibling or containment relationship in the global context of the whole XML database. Without showing such relationship, the results alone could be misleading or less comprehensible.

EXAMPLE 2. In Figure 1, a query “pencil black” will get the following results by LCA: (1) the subtree rooted at node r13(item); (2) the subtree rooted at node r17(item); (3) the subtree rooted at node r27(item). Without showing the relationships among the results, it is hard to know that the third result is completely different from the first two results because it is a make-up pencil rather than a normal pencil. If we display the results in the context of the whole XML data tree, this can be easily identified by the user.

To address the above two issues, we find that besides enhancing the core result retrieval techniques, the search engine should be able to provide an interactive way to allow users to conveniently visualize, manipulate, and organize their in-context search results.
Inspired by Google Maps with both search functionality and interactive component, we try to equip the traditional XML keyword search engine with our novel interactive Map-Like Exploration Model on XML Database (XMAP) [4], in order to address the above two issues and enhance users’ search experience.

Our major contributions in this demo paper include:

1. We address the needs for a way to easily adjust the query results and show the query results in the global context.
2. We propose a novel exploration model, XMAP, to work as a complementary component of the XML keyword search engine, in order to enhance users’ search experience.

2. SYSTEM ARCHITECTURE

The system architecture of XMAP is shown in Figure 2. All the functionalities are supported by the components running at two sides: browser end and server end.

At the browser end, it includes three components: 1) UI controller captures the operations of the user, e.g. zoom-in operation. 2) MapPainter draws the display. 3) Cache Manager loads the missing data if some data is not available locally at the browser end (cache). Each component in browser end is implemented in JavaScript.

At the server end, there are two main components: Index Constructor and Request Handler. Index Constructor constructs an R-tree liked index [4] for indexing the layers generated, so that XMAP can efficiently locate a specific region of data on a specific layer. Request Handler is a component handling all the data requests from the browser end. It will extract required area of data through the index and send them to the user.

3. BUILDING XMAP

We build XMAP following a traditional way of building a digital geographic map. Before we address the new challenges posed when XMAP is built on XML data, let us have a quick look at the features which XMAP provides:

- **Zoom in/out**: To support the zoom in/out function, the system needs to generate multiple layers of data, each of which is a complete map but has different scales of details.
- **Dynamic data loading**: The area of a user’s device where the map can be displayed is called the window. User’s device only needs to load part of the map, which is needed for display, rather than loading the whole map into the device.
- **Navigation**: When a user moves left/right/up/down the map, the missing data to be displayed in the window will be dynamically transferred from the system to the user; when a user zooms in (out), the current layer providing data to the user will be changed to a lower (higher) layer.

When implementing the above functionalities, the difference between XML data and geographic data further poses two main challenges, as discussed in the following subsections.

3.1 Challenge 1: Generating Layers for XMAP

Similar to Google Maps, we generate several copy of the same document but with different degrees of details. Every copy is called a layer. Then zooming in/out can be done by switching between different layers.

Let the number of layers be \( n \). Layer 1 is the most detailed layer; layer \( n \) is the most abstract layer and is supposed to be with a size such that it can fit into users’ window.

Therefore, layer 1 is easy to generate. We only need to present the whole XML as a tree without losing any detail on layer 1. Then every node in the XML has its own coordinates \((x, y)\) on the layer. Note that these coordinates will be the global coordinates and unchanged in other layers, because the other layers are generated based on layer 1.

Now we generate layer 2 up to layer \( n \), which is the main problem addressed in this subsection. Normally, for geographic data, these layers will be generated by scaling layer 1 with a fixed ratio, like the height:width ratio adopted in Google maps (e.g., the shape of a country should not be changed no matter which layer it is in).

However, most XML databases store millions of similar-structured data at the same hierarchical level, and keep growing along the horizontal axis. E.g., right below a node of a chain-store, the items keep growing, leading to the growing of the width of XML data tree.

Therefore, the challenge is how to scale along the dimension of width when we generate layer 2 to layer \( n \). Our solution to the challenge is to merge the compatible subtrees into groups in the XML data tree, in order to scale the layer on the dimension of width. Before introducing the concept of compatible, we will define some terms which will be used.

**Definition 1. Node Type**: The type of a node \( v \) in the XML tree, denoted as \( v\text{-type} \), is the tag name path from root node of the XML tree to \( v \).

**Definition 2. Compatible**: Given an XML data tree, two subtrees \( T_1 \) and \( T_2 \) are said to be compatible if and only if they satisfy the following conditions: 1) the roots of \( T_1 \) and \( T_2 \) are siblings; 2) the roots of \( T_1 \) and \( T_2 \) correspond to the same node type.

For example, in Figure 1, the three subtrees rooted at \( r_{13}, r_{17}, r_{21} \) (in a dash line rectangle on the right hand side) are compatible, because \( r_{13}, r_{17}, r_{21} \) are sibling of each other and they are of the same node type “company\text,en}\text{chain-store}\text{\textslash category\textslash item”.

With the concept of compatible, to generate layer \( n+1 \) from layer 1, we can merge the compatible subtrees into a group in order to hide the details and shrink the layer on the dimension of width. We will group the compatible subtrees in a bottom-up manner. Since some
subtrees will be hidden in a group, we need to show a summary of the information in a group. Here we adopt [2] to generate a snippet as a summary of the information of a group considering data representatives.

3.2 Challenge 2: Dynamical Data Loading

Since dataset could be huge, it requires the browser to load only the needed data for user’s current display rather than loading the whole XML data. Therefore, an efficient way to locate and retrieve the needed data is in demand.

Normally, R-tree [1] is used to index spatial data in a single multi-dimensional space. However, in XMAP, there are a number of closely related layers, which means there are a number of two-dimensional spaces. One possible solution is to build one R-tree index for each layer, but we find XMAP has the following two properties: P1: each layer, of XMAP is generated based on a previous layer, by merging compatible subtrees together. Therefore, amongst the layers, there exists an intrinsic hierarchical structure. P2: each layer, shrinks based on the previous layer, by a certain scale. These two properties inspire us to index all layers in one revised R-tree, where each level of the R-tree corresponds to a layer in XMAP. P1 ensures the hierarchical structure of the tree index; P2 ensures that each level of the tree index has more entries than the previous level by a certain scale. We refer interested readers to [4] for details on the index structure.

4. DEMONSTRATION

In this section, we would like to demonstrate how XMAP offers a new yet visual way for users to explore XML data, and how it greatly enhances users’ search experience by 1) providing users an easy way to make adjustment to the query results without revising and resubmitting the keyword query; 2) showing the query results in a more human-understandable way in the global context.

Figure 3 shows a screenshot of XMAP for the query “pencil black” in Example 2. As we can see, on the left hand side, it shows the results returned by existing XML keyword search methods page by page. On the right hand side, the XMAP display window works as an interactive component for users to visualize, manipulate and further explore the query results.

**XMAP Display (with Dynamically-loaded Data)**

On the right hand side of Figure 3, a XMAP display window is available to enhance users’ search experience. In the display window, users can see the XML data from a specific layer (see Section 3.1) in a map-like style. Data needed for display is dynamically loaded. For each XML node, the content of the node is shown in a 2-D rectangle, where tag names are shown in normal font and values are shown in italic font. The 3-D rectangles represent groups, each of which is a group of compatible subtrees as discussed in Section 3.1. On the surface of the 3-D rectangles, a summary of the group will be shown and the query results will also be highlighted.

Note that on the left pane, the query results are displayed page by page if there are too many results. The results on the current page will be highlighted at the XMAP display. Each query result is highlighted by an orange rectangle. The letter assigned to each result is also shown to help users distinguish the query results easily. Once user clicks on a particular result on the left pane, it will automatically take her to the corresponding subtree in the right pane (similar to Google Map).

**Addressing Motivation 1** In the XMAP display window, a dragging pad and sliding bar are provided for user to move left/right/up/down and zoom in/out, to further explore the query results and XML data. In this way, users with different search intentions can easily adjust the query results to meet their information needs without revising and resubmitting the keyword query. E.g., for the query “Allen female” in Example 1, user can easily use the dragging pad to explore the information of a cashier or the chain-store just above it.

**Addressing Motivation 2** XMAP displays the results in a global context, which makes it much easier to digest the query results. As shown in Fig. 3, for the query “pencil black”, the three pencils being returned, namely A, B and C, are not all in the same category. From XMAP display, we can easily know that result C is a make up pencil rather than a normal pencil. This is not possible to know with the traditional result list without XMAP.

5. ACKNOWLEDGEMENTS

This research was carried out at the SeSaMe Centre. It is supported by the Singapore NRF under its IRC@SG Funding Initiative and administered by the IDMPO.

6. REFERENCES