

PIX: A System for Phrase Matching in XML Documents: A Demonstration

Sihem Amer-Yahia
AT&T Labs–Research
sihem@research.att.com

Divesh Srivastava
AT&T Labs–Research
divesh@research.att.com

Mary Fernández
AT&T Labs–Research
mff@research.att.com

Yu Xu
UC San Diego
yxu@cs.ucsd.edu

Abstract

We present a system that enables flexible and efficient phrase matching in XML documents. Since XML allows structured and unstructured information to be interleaved, phrase matching in XML raises new challenges. Our system, named PIX, permits phrase matching in XML documents that contain “mixed content”. A key feature of PIX is that users can specify which element and content to ignore when matching a phrase. PIX uses inverted indices and an efficient evaluation algorithm to compute the set of matches and returns answers where phrases, ignored tags and content are highlighted. In addition, query answers are sorted using a ranking function. PIX is implemented as an extension of GALAX, a full-fledged XQuery engine. The functionality of PIX is fully integrated into XQuery and permits a natural combination of XPath-based structure matching with phrase matching.

1 Introduction

XML encourages users to insert new information into unstructured text in order to provide more semantics in a document. Augmenting unstructured text serves two purposes: *markup text* with specific tags or *annotate text* with new text that provides additional semantics. For example, the XML documents published by the Library Of Congress (LOC) [9, 14] contain large portions of text that describe actions in bills. In this text, the names of the sponsors of a bill are marked up using the tag `<sponsor>` and the content of `<footnote>`, can be inserted to provide explanatory information.

In order to query text in XML documents, existing Information Retrieval (IR) techniques for phrase matching could be used. However, indexing methods in IR assume “contiguity of keywords” in a phrase match [11]. If tags are

added to text in a document, this contiguity may be lost and phrase matching needs to account for the added tags. For example, the text “Mr. English introduced this bill which was referred to the Committee on Financial Services” is extracted from a LOC document and could be modified as follows: “`<sponsor>` Mr. English `</sponsor>` introduced this bill which was referred to the Committee on Financial Services”. The phrase “Mr. English introduced this bill” does not match the modified text anymore unless `</sponsor>` is *ignored* during the phrase match. The same phrase would not match either if the text is modified as follows: “Mr. English `<footnote>` For himself and Mr. Coyne.`</footnote>` introduced this bill which was referred to the Committee on Financial Services”. In this case, the whole content embedded in `<footnote>` . . . `</footnote>` should be *ignored* to match the phrase. Thus, in order to be able to offer flexible phrase matching in XML documents, the system should have the ability to *ignore specific tags* and *ignore specific content* when matching a phrase.

We describe PIX (Phrase matching In XML), a system for phrase matching in XML documents. In PIX, queries may specify which tags and content to ignore for phrase matching. This feature is implemented as an XQuery function that is integrated into XQuery [5]. Thus, users could ask queries that combine phrase matching with existing XQuery functionalities. For example, it is possible to ask for the paragraph that describes bills dated on June 5, 2008, sponsored by Mr. English and that contain the phrase “committee on financial services” ignoring tag `<committee-name>` and ignoring the content of `<footnote>`. PIX is implemented as an extension of the GALAX system (<http://db.bell-labs.com/galax>), a full-fledged XQuery engine.

Section 2 contains an example. Section 3 describes the architecture of PIX. Section 4 describes the demonstration. Section 5 contains some future directions.

2 Example

We consider XML documents produced by the Library Of Congress [9]. These documents are partially structured and contain large portions of annotated text. The example below describes information about a bill such as its stage, congress, session number, legislation number, location and action. An action has a date and contains text that describes its content and its sponsors. Note that the text that describes an action “is coherent” even if some tags (e.g., `<sponsor>`) are removed from the text.

```
<bill bill-stage = 'Introduction'>
  <congress> 110th CONGRESS </congress>
  <session> 1st session </session>
  <legis-num> H.R. 133 </legis-num>
  <current-chamber>
    IN THE HOUSE OF REPRESENTATIVES
  </current-chamber>
  <action date = 'June 5, 2008'>
    <action-desc>
      <sponsor> Mr. English
    </sponsor> <footnote> For himself
      and Mr. Coyne </footnote> introduced
      this bill.
    </action-desc>
  </action>
</bill>
```

We classify queries on text as follows:

Queries on text only use full-text search operators as in Information Retrieval (IR) [11]. They include word and phrase search, stemming, proximity search and prefix and suffix search. For instance, the query that finds all bills where “Mr. English” and “Mr. Coyne” are within 6 intervening words makes use of proximity search for phrase matching.

Queries on text and structure search on both text and positions of elements in the text. Several proposals [3, 4, 7] have studied techniques for combining IR searches with document structure. As an example, the query that returns (i) text elements containing both “Mr. English” and “Mr. Coyne” and (ii) its preceding and following text elements, combines keyword search with “sibling” axis.

Queries that span structure: An example is the query that returns bills containing the phrase “referred to the committee on financial services”. This query operates on the text contained in an action as a whole, ignoring the tag `<committee-name>` and the content of `<footnote>`.

We are interested in the last kind of queries since they are the most challenging and demand new indexing techniques and new evaluation algorithms.

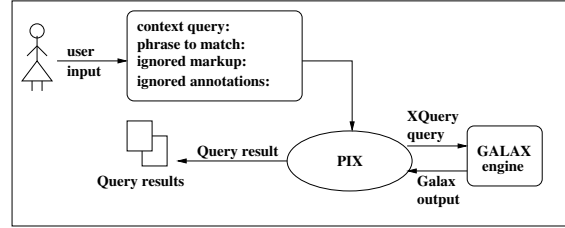


Figure 1. Architecture of PIX

3 System Architecture

Figure 1 depicts the architecture of PIX. Users can browse input XML documents before writing queries. Queries are formulated through an interface where users input the context query as an XPath expression, the phrase to match and the tags and content to ignore. The tags and content to ignore are specified as a list or as an XPath query. An XQuery query is then generated and sent to the GALAX engine. The result is displayed to the user as an HTML document in which the matched phrase, the ignored tags and ignored content are highlighted. In addition, query results are sorted using a ranking function that can be selected from the interface.

3.1 Visualizing Input Documents

Since we cannot expect all users to know the exact structure and content of each XML document, PIX allows to visualize input documents and their DTD in order to help identify which phrases to match and which tags and content to ignore when matching a phrase.

3.2 Formulating Queries in PIX

Users can input queries using a form interface in which they specify: (i) The **context query** as an XPath expression whose result is a set of elements in which the phrase is matched; (ii) The **phrase**; (iii) The **tags to ignore** as an explicit list or as an XPath query; (iv) The **content to ignore** as an explicit list or as an XPath query. The generated XQuery query is sent to the GALAX engine.

PIX supports nested matches where phrases appear inside ignored content and are identified recursively. The result of a query is a possibly empty subset of context nodes.

3.3 Indexing and Evaluation

We use a stack-based sort merge algorithm of the family of algorithms in [1]. The added novelty is the ability to match keywords in phrases and document structure according to global document order. It uses inverted lists for keyword matching and a preorder/postorder index on tags for

ignoring tags and content. The algorithm is implemented as a function in XQuery which differs from a user-defined function in XQuery (as defined in [5]) because it has to make use of inverted indices during evaluation time.

The time complexity of the algorithm (CPU and I/O) is linear in the sum of sizes of the input lists. The space complexity (size of the main memory needed to keep the intermediate stack data structure) is proportional to the nesting depth of the input XML document.

3.4 Displaying Query Results

Query results are context nodes containing at least one phrase match each. The result of a query is converted into a HTML document in which highlighting in different colors differentiates between tags, content and phrases in each context node. Users could also request ranking of results using one of three available ranking functions: (i) sort context nodes in decreasing order of the number of matches of a phrase, (ii) sort context nodes in decreasing order of the sum of scores of each phrase match in the context node and, (iii) sort context nodes in decreasing order of the maximum of scores of each phrase match in the context node. A penalty, which is a negative number equal to -1 (could be overridden by user), is assigned to each ignored tags and content. The closer are the terms in a phrase (i.e., the smaller is the number of tags and content that need to be ignored), the higher is the score of a phrase match.

4 Demonstration Overview

Aids to Query Formulation: Users can visualize and select input documents to be queried and also view their DTD. This feature helps determine which phrases to match and which tags and content might need to be ignored when matching a phrase.

Aids to Query Specification: Users need to specify an XPath query to identify context nodes as well as a phrase to match. Tags and content to ignore might be specified as an explicit list of tags or as an XPath query. If tags and content are not specified, IR phrase matching is performed. The PIX demo does not permit the user to enter arbitrary XQuery expressions but instead gives a simple form interface. Finally, the demo contains a set of sample queries that could be selected and modified by the user.

Answer Explanation: A context node qualifies as a query answer if it contains at least one occurrence of the phrase. Users might want to see “all” matches of a phrase in a context node. Thus, we adapted our algorithm to be able to identify and display all matches in a context node.

5 Future work

We are currently investigating approximate matching and scoring of query results as a new direction in PIX. We would like to combine approximate phrase querying and approximate structure querying as in [2, 3, 6, 7, 8, 12, 13].

References

- [1] S. Al-Khalifa, H. V. Jagadish, N. Koudas, J. M. Patel, D. Srivastava, Y. Wu. Structural joins: a primitive for efficient XML query pattern matching. In *ICDE*, 2002.
- [2] S. Amer-Yahia, S. Cho, D. Srivastava. Tree pattern relaxation. In *EDBT*, 2002.
- [3] R. Baeza-Yates, G. Navarro. XQL and proximal nodes. In *SIGIR*, 2000 Workshop on XML and Information Retrieval.
- [4] E. W. Brown. Fast evaluation of structured queries for information retrieval. In *SIGIR*, 1995.
- [5] S. Boag, D. Chamberlin, M. Fernández, D. Florescu, J. Robie, J. Siméon, and M. Stefanescu. XQuery 1.0: An XML query language. <http://www.w3.org/TR/xquery>.
- [6] C. Delobel and M. C. Rousset. A uniform approach for querying large tree-structured data through a mediated schema. In *Proceedings of International Workshop on Foundations of Models for Information Integration (FMII)*, 2001.
- [7] N. Fuhr and K. Grossjohann. XIRQL: An extension of XQL for information retrieval. In *SIGIR*, 2001.
- [8] Y. Hayashi, J. Tomita, and G. Kikui. Searching text-rich XML documents with relevance ranking. In *Proceedings of SIGIR Workshop on XML and Information Retrieval*, 2000.
- [9] The Library of Congress. <http://www.loc.gov/>
- [10] The Library of Congress XML Use Cases. http://www.loc.gov/crsinfo/xml/lc_usecases.html
- [11] G. Salton and M. J. McGill. *Introduction to modern information retrieval*. McGraw-Hill, New York, 1983.
- [12] T. Schlieder. Schema-Driven Evaluation of Approximate Tree-Pattern Queries. In *EDBT* 2002.
- [13] A. Theobald and G. Weikum. The Index-Based XXL Search Engine for Querying XML Data with Relevance Ranking. In *EDBT*, 2002.
- [14] United States Congress. XML and Legislative Documents. <http://xml.house.gov>.