Personalizing XML Text Search in PIMENT

Sihem Amer-Yahia[1] Irini Fundulaki[2] Prateek Jain[2,3] Laks Lakshmanan[4]

[1]AT&T Labs Research – USA
sihem@research.att.com

[2]Bell Labs – USA
{fundulaki,jain}@research.bell-labs.com

[3]IIT-Kanpur – India
prateekj@iitk.ac.in

[4]University of British Columbia – Canada
laks@cs.ubc.ca

Abstract

A growing number of text-rich XML repositories are being made available. As a result, more efforts have been deployed to provide XML full-text search that combines querying structure with complex conditions on text ranging from simple keyword search to sophisticated proximity search composed with stemming and thesaurus. However, one of the key challenges in full-text search is to match users' expectations and determine the most relevant answers to a full-text query. In this context, we propose *query personalization* as a way to *take user profiles into account* in order to *customize query answers* based on individual users' needs.

We present PIMENT, a system that enables *query personalization* by *query rewriting* and *answer ranking*. PIMENT is composed of *a profile repository* that stores user profiles, *a query customizer* that rewrites user queries based on user profiles and, a *ranking module* to rank query answers.

1 Introduction

A growing number of XML repositories such as the Library of Congress document collection [7], Medical data in XML [4], and the INEX repository [6], are being made available for search. Since XML has the ability to represent both structured and unstructured data, XML queries can combine structured search and full-text search [1, 3, 9]. XQuery Full-Text [11] is an extension to XPath and XQuery that allows both novice and expert users to express queries ranging from simple keyword search to sophisticated proximity search combined with, among others, stemming, stop words and thesaurus. However, one of the most challenging issue in XML full-text search these days is to meet user needs and return the most relevant answers using an appropriate scoring method [6]. We propose to use *personalization* as a way to tailor answers to XML full-text search queries to individual user needs and develop PIMENT, a prototype for personalizing XQuery Full-Text queries [11, 10] on XML corpuses.

Query personalization is defined as the process of taking a user query and modifying it using some information about the user to better suit the user's intent. Different users have different preferences and a different understanding of the search corpus. These differences influence the way users interpret query results. Therefore, user diversity should be taken into account to customize queries. In addition, queries can be complex. But more importantly, queries can be repetitive. The enforcement of user profiles removes the burden from the user to formulate the same (possibly complex query) every time she performs a search. In practice, personalization is used in many applications such as telecommunications [5] to direct user calls based on the context of the caller (e.g., location, time of day) and in Web search to modify the ranking of query answers[1] (e.g., by recording the URLs that users followed).

Query personalization through user profiles has different aspects that restrict or expand its applicability. Enforcing a user profile ranges from simply modifying the ranking of query answers while returning a subset of the original answers, to returning a totally different set of answers. The simplest scenario is the case where a user profile specifies *a search corpus* which will have an impact on scoring query answers since scores are usually normalized over the whole document collection [8]. In addition, a user profile may also contain *customization rules* that are used to rewrite user queries. A simple example of a customization rule, implemented in popular query engines, is to always apply a stemming or synonym modifier to expand query keywords. Customization rules become more sophisticated in the context of XML querying since they could combine conditions on both structure and keywords. For example, if a user searching XML documents in the Library of Congress collection [7], is looking for all bills that discuss education

**Proceedings of the 31st VLDB Conference,
Trondheim, Norway, 2005**

[1] *http://www.google.com.*

matters, then this query could be restricted so that it only searches documents written by the House of Representatives (because only such bills discuss education). Another example is the case where a user is searching a collection of documents describing movies in *http://www.imdb.com*, and the system enforces the condition that those movies should be playing in theaters close to the user's current location. This would correspond to an additional condition on document structure since theater location is represented as a sub-element of movie elements in the IMDB collection.

We propose to demonstrate PIMENT, a system for personalizing text search in XML based on pre-specified user profiles. Profiles in PIMENT can be defined for individual users. Users can configure their profile and choose to enforce it or not during query evaluation. Both queries and profiles are expressed in XQuery Full-Text which enables their interoperability and facilitates query customization. Moreover, PIMENT does not implement all possible customization scenarios but the fact that profiles are expressed in XQuery Full-Text, makes it extensible. To the best of our knowledge, PIMENT is the first prototype that provides the ability to personalize XML full-text search queries.

Section 2 describes the components of PIMENT. Section 3 provides motivating examples. Demonstration steps are given in Section 4.

2 System Architecture

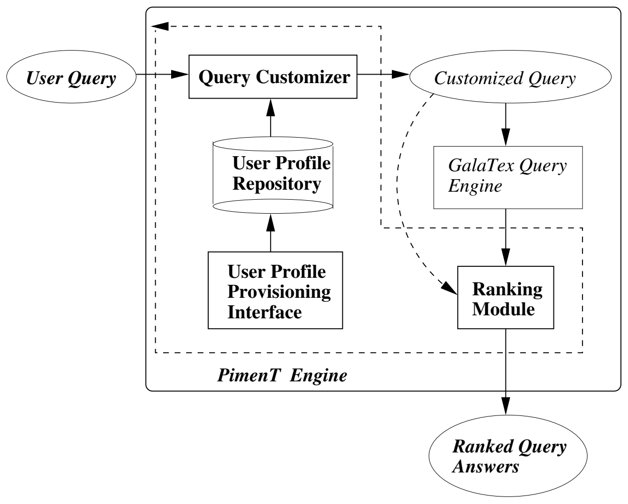

Figure 1: Architecture of PIMENT

Figure 1 depicts the architecture of PIMENT. At its core is the *Query Customizer Module*, the *User Profile Repository* and the *Ranking Module*. The *User Profile Repository* stores user profiles, where each profile is (a) associated with a domain of interest and (b) consists of a set of *customization rules*. A customization rule is of the form *(condition, action, conclusion)* where the *condition* and *conclusion* parts of the rule are XQuery Full-Text expressions and *action* can be one of *add, remove, replace*. In order to specify her profile, a user can select one domain of interest, associated with one or more XML documents. With the help of the *User Profile Provisioning Interface*, the user can manage her profile (e.g., create new rules, delete or modify existing rules). Through the query interface, the user can select a domain of interest and formulate her queries which are then processed by the *Query Customizer*. This module (i) retrieves from the user profile repository the profiles (and hence the rules) that are relevant to the query and (ii) rewrites the query using the retrieved rules. The result of this process is a *customized* query expressed in XQuery Full-Text, and consequently can be evaluated using any XQuery Full-Text engine[2]. PIMENT is built on top of GALATEX [2], a conformant implementation of XQuery Full-Text[3]. Finally, answers are returned sorted by their relevance to the user's query and profile.
In order to reflect the application of user profile rules to a query, ranking query answers may be enforced outside the XQuery Full-Text engine which requires the *Ranking Module* of PIMENT to resort some query results. Profile rules can be applied in different orders which may result in different customized queries and hence different query answers for the same initial user query. We propose to experiment with different rule application strategies.

3 Motivating Examples

In this section, we show how the application of user profiles may alter the set of answers to a user query in order to better meet user's preferences. All queries and user profile rules in our examples are expressed in XQuery Full-Text, an extension to XQuery to support full-text search. More precisely, queries and user profiles are expressed in the XPath subset of this language. XQuery Full-Text is based on TeXQuery [1] that defines two new XQuery expressions FTContainsExpr and FTScoreExpr. It supports all full-text search primitives referred to as FTSelections in Figure 2. FTSelections are fully composable and are based on the *AllMatch* data model. Due to space limitations, we refer the reader to [1, 11] for more details on the language.

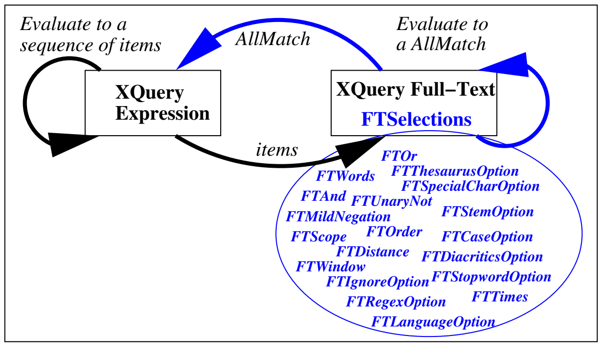

Figure 2: XQuery and XQuery Full-Text Composability

We classify profile rules into *generic rules* that apply to

[2]Example implementations are *Quark, XIRQL, XXL, ApproXML, ELIXIR, PwerDB-IR, JuruXML, XSearch* and *Timber.*

[3]*http://www.galaxquery.org/galatex.*

a query regardless of its content and *query-specific rules* that apply to a query if it satisfies some conditions. We use examples from the Library of Congress [7], and from the MERIMEE collection[4] of descriptions of buildings from the French Ministry of Culture.

3.1 Library of Congress

<*resolution*>
  <*comment*>.. human rights violations .. prisons< /*comment*>
  <*comment*>.. violations of human rights ..< /*comment*>
  <*comment*>.. animal rights .. human .. violate ..< /*comment*>
  <*comment*>.. violate the human rights ..prisons < /*comment*>
  <*comment*>.. Human rights ... violations < /*comment*>
< /*resolution*>

Figure 3: A simplified XML document from the LoC

Generic rules: Consider a novice user U_1 who is interested in US Congress resolutions and whose customization rule specifies that the options *stemming* and *case insensitive* should apply to all query keywords. On the other hand, an expert user U_2 who knows exactly what she is looking for, specifies in a rule that *"without stemming"* and *"case sensitive"* options should always apply unless otherwise instructed in the query. These user rules are shown in Table 1.

R_1:	((ftcontains ∗), add, (ftcontains ∗ with stemming case insensitive))
R_2:	((ftcontains ∗), add, (ftcontains ∗ without stemming case sensitive))

Table 1: Rules for users U_1 and U_2

Consider that the two users search resolutions on violations of human rights using the query:

q: *//comments* ftcontains 'human' && 'rights' && 'violations'

Applying rule R_1 to q results in query q_1 shown in Table 2. One can observe that the *"with stemming"* and *"case insensitive"* options are added in the rewritten query. Similarly, after enforcing rule R_2 of user U_2, we obtained q_2 (also shown in Table 2) to which the options *"without stemming"* and *"case sensitive"* have been added. Note that customization rules override whatever stemming and case sensitivity defaults the underlying query engine has.

q_1:	*//comments* ftcontains ('human' && 'rights' && 'violations') with stemming case insensitive
q_2:	*//comments* ftcontains ('human' && 'rights' && 'violations') without stemming case sensitive

Table 2: Rewritten queries q_1 and q_2 for query q using the rules R_1 and R_2 resp.

One can observe that the user rule R_2 restricts the set of answers that would have been obtained if the initial user query q was evaluated on the simplified document in Figure 3: the first and second *comment* elements depicted in a box in Figure 4 are returned as answers to q_2.

<*resolution*>
  $\boxed{\text{<comment>.. human rights violations .. prisons< /comment>}}$
  $\boxed{\text{<comment>.. violations of human rights ..< /comment>}}$
  <*comment*>.. animal rights .. human .. violate ..< /*comment*>
  <*comment*>.. violate the human rights ..prisons < /*comment*>
  <*comment*>.. Human rights ... violations < /*comment*>
< /*resolution*>

Figure 4: Answers to query q_2

Query-specific rules: Consider now another user U_3 who specifies that whenever a search contains the terms *'human'*, *'rights'* and *'violations'*, the first two terms must appear as a phrase, *stemming* should be applied to the term *'violations'* which should appear before or after the phrase *'human rights'* with a distance of at most 2 words (without considering stop words). This rule is shown in Table 3.

R_3:	((ftcontains 'human' && 'rights' && 'violations'), replace, (ftcontains ('human rights' && 'violations' with stemming) ordered without stop words within window 2))

Table 3: Rule for user U_3

Consider query q' that requests US Congress resolutions on violations of human rights in prisons:

q' : *//comments* ftcontains 'human' && 'rights' && 'violations' && 'prisons'

Applying R_3 to q' results in query q_3 shown below whose answers are restricted to *comment* elements shown in a box in Figure 5.

q_3:	*//comments* ftcontains ('human rights' && 'violations' with stemming) ordered without stop words within window 2 && 'prisons'

<*resolution*>
  $\boxed{\text{<comment>.. human rights violations ..prisons< /comment>}}$
  <*comment*>.. violations of human rights ..< /*comment*>
  <*comment*>.. animal rights .. human .. violate ..< /*comment*>
  $\boxed{\text{<comment>.. violate the human rights .. prisons< /comment>}}$
  <*comment*>.. Human rights ... violations < /*comment*>
< /*resolution*>

Figure 5: Answers to query q_3

3.2 Cultural example (MERIMEE)

Generic rules: Consider an archaeologist A_1 who queries the MERIMEE document of Figure 6. A_1 is interested mainly in buildings that are found in Alsace. A_1's user profile contains two rules; one that requires that whenever she asks for descriptions of buildings, the structural predicate *//building[placeName* ftcontains *'Alsace']* should be added to the query, and also that the search should be expanded to the subregions of Alsace (e.g. Enzheim) as specified by the Getty thesaurus of geographical names (TGN)[5]; another

[4] *http://www.culture.gouv.fr/documentation/merimee/accueil.htm.*

[5] *http://www.getty.edu/research/conducting_research/vocabularies/tgn/.*

rule, that specifies that whenever a reference to "concrete" is specified, this should be replaced with a condition on the building's period. These user rules are shown in Table 4.

```
<buildings>
  <building>
    <placeName>Enzheim< /placeName>
    <descr>.. facade .. brick .. clay ..< /descr>
    <period>renaissance< /period>
  < /building>
  <building>
    <placeName>Alsace< /placeName>
    <descr>.. facade .. sculpture.. concrete ..< /descr>
    <period>modern 1970's with a medieval touch< /period>
  < /building>
  <building>
< /buildings>
```

Figure 6: An example document from MERIMEE

R_4:	(∗, add, (*//building[placeName* ftcontains 'Alsace' with thesaurus TGN]))
R_5:	((*descr* ftcontains 'concrete'), replace, (*period* ftcontains 'modern'))

Table 4: User profile for archeologist A_1

Assume that user A_1 asks the query

q: *//building[descr* ftcontains *'concrete']*

Given the user profile shown in Table 4, q is rewritten to q' given in Table 5. The answer to q' is the second *building* element of the document shown in Figure 6.

q':	*//building[placeName* ftcontains 'Alsace' with thesaurus TGN and *period* ftcontains 'modern']

Table 5: Rewritten query q'

Query-specific rules: Consider another archaeologist who is also interested in buildings found in Alsace:

q: *//building[placeName* ftcontains 'Alsace']

Her profile specifies that for such a query, the search should be *(i)* expanded to all the regions of Alsace, *(ii)* restricted to the buildings that were built during the renaissance period (i.e. the predicate *//building[period* ftcontains *'renaissance']* should be added in the query), and *(iii)* the description of the building must contain information on its facade and a reference to some material. This rule is illustrated in Table 6 (where the term 'materials hierarchy' is the root term of the *materials hierarchy* of the *Art & Architecture Thesaurus*[6]).

Query q is thus rewritten to query q' which is similar to the conclusion of the user rule R_6. The result for query q contains the *second* building element of the document in Figure 6 whereas, the result for query q' contains the *first* building element. One can observe how a very simple query can be rewritten very easily to a more complex one by using the user profile rules.

[6] *http://www.getty.edu/research/conducting_research/vocabularies/aat/.*

R_6:	((*//building[placeName* ftcontains 'Alsace']), replace, (*//building[placeName* ftcontains 'Alsace' with thesaurus TGN and *period* ftcontains 'renaissance' and *descr* ftcontains 'facade' && ('materials hierarchy' with thesaurus AAT)]))

Table 6: User rule R_6

4 Demonstration Overview

Aids to Query Formulation: Users can select various documents from the LoC, and MERIMEE collections among others.
Aids to Profile Provisioning: An interface is provided to create and update user profiles. PIMENT validates user profiles by checking conflicts between the rules it contains.
Query Customization: We provide the ability to experiment with different profile application strategies (e.g., apply *add* rules, then *remove* rules and finally, *replace* ones).
Answer Explanation: We show how modifying profiles may change the set of returned answers, their number and their ranking.

References

[1] S. Amer-Yahia, C. Botev, J. Shanmugasundaram. TeXQuery: A Full-Text Search Extension to XQuery. WWW 2004.

[2] E. Curtmola, S. Amer-Yahia, M. Fernández, P. Brown. GalaTex: A Conformant Implementation of XQuery Full-Text. XIME-P 2005.

[3] N. Fuhr, K. Grossjohann. XIRQL: An Extension of XQL for Information Retrieval. SIGIR Workshop on XML and Information Retrieval, 2000.

[4] Health Level Seven. *http://www.hl7.org.*

[5] R. Hull, B. Kumar, D. Lieuwen, P. F. Patel-Schneider, A. Sahuguet, S. Varadarajan, A. Vyas. Enabling Context-Aware and Privacy-Conscious User Data Sharing. MDM 2004.

[6] Initiative for the Evaluation of XML Retrieval. *http://inex.is.informatik.uni-duisburg.de:2004/.*

[7] Library Of Congress Documents in XML. *http://www.loc.gov/crsinfo/xml.*

[8] G. Salton, M. McGill. Introduction to Modern Information Retrieval. McGraw-Hill, 1983.

[9] A. Theobald, G. Weikum. Adding Relevance to XML. WebDB 2000.

[10] The World Wide Web Consortium. XQuery 1.0 and XPath 2.0 Full-Text Use Cases. *http://www.w3.org/TR/xmlquery-full-text-use-cases/.*

[11] The World Wide Web Consortium. XQuery 1.0 and XPath 2.0 Full-Text. W3C Working Draft. *http://www.w3.org/TR/2005/WD-xquery-full-text-20050404/.*