FXPath: Flexible Querying of XML Documents

Daniele Braga[1] **Alessandro Campi**[1] **Ernesto Damiani**[2] **PierLuca Lanzi**[1] **Gabriella Pasi**[3]

[1]Politecnico di Milano, Dipartimento di Elettronica e Informazione
[2]Università di Milano, Dipartimento di Tecnologie dell'Informazione
[3]Istituto per le Tecnologie della Costruzione, Consiglio Nazionale delle Ricerche

{braga,campi,lanzi}@elet.polimi.it edamiani@crema.unimi.it gabriella.pasi@itim.mi.cnr.it

Abstract

In this paper a fuzzy extension of the XML XPath language is proposed. The aim is to increase flexibility while retaining full backward compatibility with the existing standards.

Keywords: XML, XPath, fuzzy, flexibility, semi-structured data.

1 Introduction and Motivations

XML is increasingly gaining importance as a standard data format for information interchange on the WWW. The XML data-model, called Infoset (`www.w3.org/TR/xml-infoset`), defines XML data as *multi-sorted trees*, i.e. trees whose nodes belong to different types. In the following example (Fig. 1) we show a sample XML tree including element, attribute and content nodes (the actual XML Infoset has more types).

XPath 1.0[1] (`www.w3.org/TR/xpath`), is a standard language that allows to write "tree traversal" expressions for selecting XML tree nodes. Such tree traversal expressions have been adopted since long as a World Wide Web Consortium (W3C) Recommendation and are used in several XML-related applications, together with other standards like *XPointer* (`www.w3.org/TR/xptr`) and *XSLT* (`www.w3.org/TR/xslt`); some flexibility support is obtained in XPath by means of wildcards [5]. XPath expressions are also used as selection conditions in the framework of fully-fledged XML query languages. The database community has proposed several query languages for XML, some of them as a development of previous languages for querying semi-structured information[2].

```
<visitcard>
  |
  |----- <personaldata id="0124">
  |        |
  |        |---- <name>
  |        |       |----- John
  |        |     </name>
  |        |
  |        |---- <surname>
  |        |       |----- Smith
  |        |     </surname>
  |        |
  |      </personaldata>
  |
  |----- <companydata>
  |        |
  |        |----- <name>
  |        |        |----- XMLSolutions
  |        |      </name>
  |        |
  |        |----- <position>
  |        |        |----- VicePresident
  |        |      </position>
  |        |
  |      </companydata>
  |
</visitcard>
```

Figure 1: A Sample XML tree

In the last few years, much work has been done towards a standard for XML querying and recently the W3C endorsed the XQuery language (`www.w3.org/TR/xquery`) as a candidate recom-

[1]Version 2.0 of the XPath language is currently being defined, but is not yet stable enough to be taken as a basis for extensions.

[2]Two detailed comparisons (both involving four languages) can be found in [1] and [5]

mendation. Both in XQuery and XPath a retrieved information item is usually a *node set*. Traversal expressions may include conditions and wildcards. For example, the following XPath expression

```
document("visitcard.xml")/visitcard/*/name
```

produces as a result a set containing two `name` element nodes, while the expression

```
document("visitcard.xml")/personaldata[@id =
    "0124"]
```

returns the singleton set containing exactly one `personaldata` element node. The square brackets are used to express conditions that filter the nodes retrieved by the expression they are attached to, and the `@` qualifier denotes that `id` is an attribute node.

The main advantages of XPath include:

- a rich set of available built-in expressions;
- elementary data-types: Boolean, Number, String, Node set;
- specification of selection conditions to be satisfied by relevant nodes;
- availability of extendable utility functions.

2 FXPath: fuzzy extensions to XPath

In this section we introduce our proposed extension to XPath, taking into account the following characteristics:

- *Fuzzy Subtree Matching*: providing a ranked list of retrieved information items rather than the usual set oriented one.
- *Fuzzy Predicates*: specifying flexible selection conditions.
- *Fuzzy Quantification*: allowing the specification of linguistic quantifiers as aggregation operators.

2.1 Ranked-list of retrieved information items

XPath 1.0 (and, as a consequence, XQuery) inherited two main features, shared by most previous XML query languages, that are relevant to our approach:

- *Path-based selection*, based on the assumption that the user knows enough of the target schema to be able to formulate a *search path* that can be matched against the structure of the target XML documents. As we have seen, search paths are usually given in the standard form of *XPath* expressions.
- *Set-oriented query result*: the path-based selection mechanism retrieves portions of XML documents, namely the set of nodes matching the user-provided path. All retrieved portions equally belong to the query result set, even when the query exploits the facilities provided by the language for partial or flexible pattern matching.

With respect to the first feature, we observe that when querying non-validated or heterogeneous XML data, the assumption that the user is fully aware of the target schema is debatable, because the users have no single schema to rely on in order to write the query graph's structure. Often, all users have at their disposal is a sample document, or at most a tag repertoire, i.e. the union of the XML *namespaces* used throughout the XML document base. In this situation, trying to find an exact match of the query pattern to a part of the target document is likely to result in *silence*, as the query topology, however similar, will probably not match the document's structure.

Regarding the second feature, we remark that approximate queries hardly ever intend to dictate the exact structure of the query result; rather, they provide a loose example of the information the user is interested in. Therefore several degrees of matching should be possible. This situation is not unlike the one encountered in the field of multimedia databases [8], where query results are typically ranked lists, ordered according to some similarity measure.

2.2 Fuzzy extensions

In this Section we describe an extension to the syntax and semantics of XPath tree traversal expressions, aimed at supporting flexibility and

ranked results. Flexibility is here intended as the capability of dealing with imperfect (imprecise/vague) information and is modelled at two distinct levels.

- the level of expression of soft selection conditions by means of *fuzzy predicates*;
- the level of expression of complex conditions by aggregating selection conditions by means of *linguistic quantifiers*.

2.2.1 Fuzzy predicates

Traditional query languages (in the database domain) allow data selection based on binary predicates. The evaluation of a binary predicate produces a clear-cut partition of the target data, those which fully satisfy the condition (the relevant ones), and those which do not. An example is offered by the request of people having an age over a specified threshold. As a consequence, data relevance with respect to a query is modelled as a binary concept: either an information item is relevant or it is not. On the other hand, a *vague* predicate, represented by a fuzzy set, expresses a "softer" condition. A vague predicate is a fuzzy subset of the attribute domain (the latter being a usual, i.e. a crisp set of values); the definition of the membership function is given in accordance with the semantics of the linguistic term that expresses the fuzzy predicate. In the case of age, for instance, a condition expressed by a fuzzy set such as `young` specifies a softer constraint than the one specified by a numeric threshold. Vague conditions can be satisfied to a certain extent, allowing for ranking the results of query evaluation [4].

In order to allow for vague specification of selection conditions of XML nodes, thus obtaining a ranked list as a result of a XPath query, we propose to admit the specification of fuzzy predicates in XPath. In the following we shall refer to our extended version of XPath as `FXPath`. In order to keep our new language features separate from the standard specification, we enclose non-standard expressions into curly braces, so that a simple parsing algorithm can translate it into the corresponding crisp expression. An FXPath expression can contain fuzzy conditions, and each of them is satisfied by XML items to a degree in $[0, 1]$. We shall rank FXPath query results according to these degrees.

2.2.2 Linguistic Quantifiers

Linguistic quantifiers extend the set of quantifiers of classical logic. They can be either crisp (such as for example `all`, `at least 1`, `at least k`, `half`) or fuzzy (such as for example `most`, `several`, `some`, `approximately k`).

Two kinds of fuzzy quantified propositions can be distinguished: "Q X are Y", i.e., Q elements of set X satisfy the fuzzy predicate Y, and "Q B X are Y", i.e., Q elements of set X which satisfy the fuzzy predicate B also satisfy the fuzzy predicate Y. Examples of such statements are: most of the selection conditions are satisfied and most of the important selection conditions are satisfied.

Formally, fuzzy quantifiers have been first defined as fuzzy subsets and are of two main types: absolute and relative. Absolute quantifiers, such as `about` 7, `almost` 6, etc. can be defined as fuzzy sets with membership function $Q : R \longmapsto [0, 1]$, where $Q(x)$ indicates the degree to which the amount x satisfies the concept Q. Relative quantifiers like `most` are defined as fuzzy subsets of the unit interval: $Q : [0, 1] \longmapsto [0, 1]$, where $Q(x)$ indicates the degree to which x satisfies the concept Q. Other formalizations of linguistic quantifiers have been defined, among which the *Ordered Weighted Averaging* operators (OWA). This formalization allows the association of linguistic quantifiers with mean aggregation operators, and it is well suited in the context of information systems.

In the Boolean query language, the AND and OR connectives allow a crisp aggregations of selection conditions, and correspond to the linguistic quantifiers all and at least 1 respectively. The `AND` used for aggregating M selection conditions does not tolerate the unsatisfaction of a single condition; this may cause the rejection of useful items. Within the framework of fuzzy set theory some generalizations of the Boolean query language have been defined based on the concept of linguistic quantifiers: they are employed to specify both crisp and vague aggregation criteria of the selection conditions [2, 3].

```
<articles>
   <article year="2001" num_pages="8">
      <title>t1</title>
   </article>
   <article year="2001" num_pages="12">
      <title>t2</title>
   </article>
   <article year="2002" num_pages="10">
      <title>t3</title>
   </article>
</articles>

<articles>
   <2001>
      <article num_pages="8">
         <title>t1</title>
      </article>
      <article num_pages="12">
         <title>t2</title>
      </article>
   </2001>
   <2002> ... </2002>
</articles>
```

Figure 2: Two XML fragments

New aggregation operators can be specified by linguistic expressions, with a self-expressive meaning such as *at least k* and *most of*. They are defined with a behavior between the two extremes corresponding to the `AND` and the `OR` connectives, which allow, respectively, requests for all and at least one of the selection conditions. By adopting linguistic quantifiers, the requirements of a complex Boolean query are more easily and intuitively formulated. In the following, we shall use linguistic quantification in the framework of flexible XPath queries.

2.3 Flexible selection techniques

One of the typical features of XML data is the difficulty in distinguishing between data and their structure. As an example, consider the two little XML fragments in Fig. 2.

These two fragments contain exactly the same information, but in the first one the information about the year of publication of the articles is represented as an attribute, in the second fragment it is represented within the structure.

This overlap between data and their structure is an intrinsic feature in semi-structured data. Since atomic information items are clustered in a hierarchical structure, we can state as a general principle that the nearer two items are, from a "physical" viewpoint, the more likely they are semantically related. This assumption does not hold for other data representation techniques, such as relational tables. Moreover, semi-structured data are typically dealt with in heterogeneous scenarios, as the above example demonstrates.

The arguments summarized above justify the need for extending XPath with flexible *structure* querying capabilities. We identified a (small) set of useful constructs, with which we believe we can intuitively express most approximate conditions, namely `NEAR`, `ABOUT`, `BESIDES`, and `LIKE`.

These keywords are intended to be somehow "polymorphic", in that they can be used in different contexts to express rather different conditions (on values, structure and cardinalities). An example of the use of `NEAR` follows.

Suppose that we need to find articles that were published as close to year 2000 as possible. Crisp XPath queries (that exactly extract documents published in year 2000) could address the first or the second fragment of Fig. 2 respectively:

```
/articles/article[@year = 2000]
/articles/2000/article
```

A first way to release the constraint would be, for the first fragment, to use a fuzzy predicate `NEAR` 2000:

```
/articles/article[{@year NEAR 2000}]
```

while for the second fragment the query would be:

```
/articles/article[{../tagname() NEAR 2000}]
```

where the expression `../tagname()` has the effect of extracting the name of the element that contains each `article` element. This name is compared to the value `2000` by the `NEAR 2000` predicate.

All previous examples were basically targeted to the comparison of values. Let us now demonstrate how fuzzy conditions can be imposed upon the *tree structure* of XML data. XPath provides two crisp constructs that can express topology constraints, as follows:

```
/articles/*/article
/articles//article
```

In the former case the * axis matches any tag in a specified position, disregarding its name (the position is known, the name is unknown), while in the latter case the // axis matches any path (even one of length zero) that descends the containment hierarchy, and all the `article` elements are matched, independently of their "distance" from the `articles` element they are contained in (the name is known, the position is unknown). If we want the degree of matching to be stronger for those elements that are nearer, we can use the keyword `NEAR` as a new XPath axis to expresses this requirement, as follows:

```
/articles{/NEAR}/article
```

The output node set will be ranked with respect to the increasing number of steps to be descended.

Note that the keyword `NEAR` is used in the previous examples with a different meaning, but in all cases it expresses 'proximity'[3].

Another kind of "proximity" we are interested in is that of the horizontal distance between siblings. This extension is based on the assumption that the elements within a list are often ordered with respect to precise criteria, and that the user might want to intuitively and synthetically formulate such a constraint[4].

For example, assuming that in the second of the fragments of Fig. 2 the years are listed in ascending order, the task of retrieving the publications published around year 2000 could be expressed as follows:

```
/article/2000{/BESIDES}/article
```

3 Ranking query results

In this section we provide some more examples of FXPath expressions and demonstrate the ranking of the query results. We begin showing the output of the previous query

```
/articles/article[@year NEAR 2000|v1]
```

where `NEAR 2000` is the usual fuzzy predicate, and the variable $v1$ (which takes a real value in the $[0, 1]$ interval) is bound to the degree of satisfaction (retrieval status value) of the condition by each XML item returned by the XPath expression. Returned items are wrapped into couples of annotations, so as to recall XML tagging:

```
<!-- RankingDirective RankingValue=".8" -->
  <article year="2001" num_pages="8">
    <title>t1</title>
  </article>
<!-- /RankingDirective -->
<!-- RankingDirective RankingValue=".8" -->
  <article year="2001" num_pages="12">
    <title>t2</title>
  </article>
<!-- /RankingDirective -->
<!-- RankingDirective RankingValue=".65" -->
  <article year="2002" num_pages="10">
    <title>t3</title>
  </article>
<!-- /RankingDirective -->
```

Here only the ranking value is shown, but much more information about the processing could enrich the annotations.

The variables bound to the degree of satisfaction of the conditions can be used in a `WITH RANKING` clause in order to combine the bound values in the final ranking. If not specified, a default policy is adopted.

The following example shows a FXPath expression with two fuzzy conditions:

```
/articles/article[{@page_num NEAR 10|v1} AND
  {@year NEAR 2000|v2}]{WITH RANKING v1 * v2}
```

The ranking of the result set is obtained as a combination of the values bound to $v1$ and $v2$.

More complex `RANKING` clauses could involve:

- Linear combination of the values bound to the ranking variables:
  `WITH RANKING 0.4*v1 + 0.6*v2`
- Normalized weighted average

[3] Recalling the first examples with the `NEAR` keyword, we can think of combining the two styles to express a more general task. If we ignore the structure of both fragments, and only have partial information about what they represent, we can express the task within a single *blind query* [6], such as `//article[{NEAR 2000}]`.

[4] XPath and, as a consequence, XQuery is particularly unsuited to express this kind of constraint, that requires an awkward use of the function `position()`.

- Ordering:
  `WITH RANKING Priority(v1,v2,v3)`

4 Conclusion and future work

Our approach is a step forward with respect to the selection techniques currently used by XML query languages, as the latter rely on exact structure matching, and therefore ignore potentially useful information. Besides dealing with structure, however, any approximate matching technique needs to assess the degree of equivalence of the information carried by two nodes or subtrees. As we have seen, our approach covers two nodes having different tags contain "more or less" the same content, but other forms of hidden equivalence may take place and should be reconciled or tolerated by means of flexible and approximate *smushing* techniques[5]. Suppose for instance that one node of an XML tree contains a direct link to a resource, while another node of a different tree contains an indirect link (with multiple pointers) to the same resource. Obviously, such node pairs are not crisply equivalent, but still they should be treated as equivalent to a degree. We plan to explore this subject in a future paper.

Acknowledgements

The authors wish to thank Stefano Ceri and Letizia Tanca for the useful discussions on approximate XML querying.

References

[1] A. Bonifati, S. Ceri, "Comparison of XML Query Languages", in: SIGMOD Record 29(1) (2000).

[2] G. Bordogna, G. Pasi, "Linguistic aggregation operators in fuzzy information retrieval", in: International Journal of Intelligent systems, 10(2), 233-248.(1995).

[3] G.Bordogna, G. Pasi, "Modelling Vagueness in Information Retrieval", in: Lectures in Information Retrieval, M. Agosti, F. Crestani and G. Pasi (eds.), Springer Verlag (2001).

[4] P.Bosc, H. Prade, "An Introduction to the Fuzzy Set and Possibility Theory-based Treatment of Flexible Queries and Uncertain or Imprecise Databases", in: Uncertainty Management in Information Systems, A. Motro, P. Smets (eds.), Kluwer Academic Publishers, pp. 285-324 (1997).

[5] S. Cluet, A. Deutsch, D. Florescu, A. Levy, D. Maier, J. McHugh, J. Robie, D. Suciu, J. Widom. "XML Query Languages: Experiences and Exemplars" `www-db.research.bell-labs.com/user/simeon/xquery.html`

[6] E. Damiani, L. Tanca, F. Arcelli-Fontana, "Fuzzy XML Queries via Context-based Choice of Aggregations", in: Kybernetika 16(4), (2000).

[7] E. Damiani, L. Tanca, "Blind Queries to XML Data", *DEXA 2000*, Greenwich (UK), LNCS 1873, pp. 345-356 (2000).

[8] R. Fagin, "Combining fuzzy information from multiple systems" *PODS'96*, Montreal (Canada), pp. 216–226 (1996).

[5] For the original definition of the problem by Dan Brinkley, see `http://lists.w3.org/Archives/Public/www-rdf-interest/2000Dec/0191.html`.