International Journal of Computational Intelligence Systems, Vol. 9, Supplement 1 (2016), 95-103

Fuzzy Approaches to Flexible Querying in XML Retrieval

Stefania Marrara and Gabriella Pasi

Department of Informatics, Systems and Communication,
University of Milano Bicocca,
Viale Sarca 336,
20126, Milan, Italy
stefania.marrara@disco.unimib.it
pasi@disco.unimib.it

Received 2 December 2015

Accepted 12 March 2016

Abstract

In this paper we review some approaches to flexible querying in XML that apply several techniques among which Fuzzy Set Theory. In particular we focus on FleXy, a flexible extension of XQuery-FT that was developed as a library on the open source engine Base-X. We then present PatentLight, a tool for patent retrieval that was developed to show the expressive power of Flexy.

Keywords: XML Retrieval, flexible language, Fuzzy Set Theory

1. Introduction

With the diffusion of the World Wide Web the information available on-line has been constantly increasing, and consequently the need for effective systems that allow an easy and flexible access to information is a crucial issue[1]. By flexibility is here meant the capability of the system to both manage imperfect (vague and/or uncertain) information, and adapt its behaviour to the user context.

To this aim Fuzzy Set Theory has been applied to Information Retrieval (IR) to the purpose of providing retrieval techniques capable of modelling, at some extent, the human subjectivity for estimating the partial relevance of documents to the user needs. An overview of how fuzzy set theory has been applied to the aim of designing flexible Information Retrieval Systems has been presented in[2].

Within Information Retrieval, XML retrieval constitutes an interesting field of research. XML[3] is the acronym of *eXtensible Markup Language*, a text format established in 1998 to the purpose of exchanging data on the Web. XML specifies a set of rules for structuring the informative content (data or text) of documents: both the structure and the content are described in plain text that can be easily read by computer programs, while remaining human-readable. The XML formatting rules allow to graphically represent documents in a tree form.

An example of fragment of XML document is shown in Fig.1. In the document fragment the structure, i.e. the nodes in the tree representation, is described by means of *element tags*. The content of each node is always confined between a *starting tag* (for instance <`author`>) and a *closing tag* (e.g., < /`author`>). *Attributes* can be used to specify meta-information related to the document elements (e.g., `genre`). The content is usually stored in leaf nodes (e.g., Umberto Eco).

The expressiveness of this language and its

adaptability to different domains of application have motivated the generation of huge collections of highly structured XML documents, and, as a consequence, the definition of ad-hoc query languages.

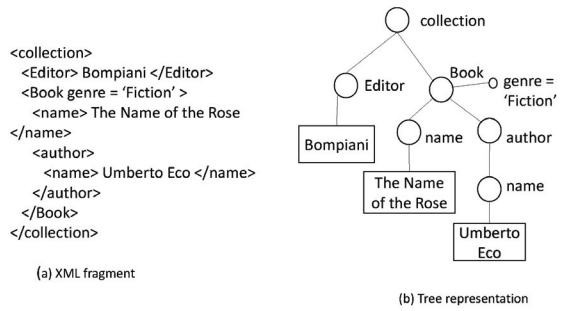

Fig. 1. A simple XML document fragment (a) and the corresponding tree representation (b)

The two main standard languages that have been defined by the World Wide Web Consortium (W3C)[6] to query XML data are XPath[7] and XQuery[8]. Both languages were initially conceived to retrieve fragments in complex documents with huge tree structures, deeply nested, but they did not provide features for evaluating keyword based queries on large textual sections. To address this issue, both XPath and XQuery have been extended by keyword based search capabilities following an Information Retrieval approach, and a new set of content related constraints has been defined. These constraints allow a full-text search with a computation of a relevance score for each XML fragment. These extensions have given birth to the W3C standard languages XPath Full Text (XP-FT) and XQuery Full Text (XQ-FT)[9]. Concerning the constraints on the document structure, XP-FT and XQ-FT allow the specification of flexible constraints (i.e., the descendant axis); however the evaluation of these constraints is "hard" as it only acts on the selection of fragments without ranking them.

This has stimulated in recent years a wealth of research aimed at improving the expressiveness of these query languages w.r.t. structural constraints, in order to provide an approximate structural matching with the consequent ranking of the retrieved fragments[4,5]. In this paper we review some approaches that have proposed an extension of both XQuery and XQuery Full Text by means of flexible constraints that are evaluated in an approximate way (section 2); in section 3 we focus on FleXy[10,11], a flexible extension of XQuery-FT initially defined in[12] and developed as a library on the open source engine Base-X[11]. In section 4 we present PatentLight[13,14], a tool for patent retrieval that was developed to show the expressive power of FleXy. Finally the conclusions are presented in section 5.

2. Related Work

In this section, after a short description of the main features of XML query languages, a review of some approaches that have been proposed to address flexibility in XML retrieval by using several techniques among which Fuzzy Set Theory is presented.

2.1. XML Query Languages: an overview

In this section some of the most important languages for querying XML documents are described. XPath[7] is the first language defined by the W3C to allow the selection of fragments of an XML document. The main component of the language is the *path expression* that allows to formulate constraints whose evaluation produces a set of matching XML fragments.

For instance, given the fragment in Figure 1, a simple XPath query retrieving the content "Umberto Eco" is `Book/author/name`. The query evaluation produces the fragment
`<name> Umberto Eco </name>`.
XQuery[8] was defined by the W3C as an extension of XPath aimed to include functionalities able to reorganize, transform and finally return a set of XML fragments in a structured form.

Since more and more companies are adopting XML as a standard, this has pushed to integrate XQuery as the main query language in their systems.

Both XPath and XQuery were created having in mind XML as a format for data organization and storage like in Database Managemen Systems. The issue of using XML as a format to define structured text was subsequently addressed by the Information Retrieval community, which aimed to enhance XML

retrieval with IR techniques for keyword-based selection of XML fragments.

To this aim, Trotman et al. proposed the NEXI[15] language: NEXI enhances the XPath selection syntax by adding the `about` constraint, to express content related constraints in a keyword-based IR style. An example of NEXI query on the fragment in Fig.1 is `Book/author/name[about(''Eco'')]`.

Due to both the increasing popularity of XPath and XQuery, and the creation of huge collections of XML documents, the XML community proposed standard languages including keyword-based query constraints. The definition of the standard XQuery Full Text and XPath Full Text[9] language extensions by the W3C includes features such as element relevance computation (element scoring), and full-text search capabilities. Any of the two standard languages however provides any flexibility in the definition of structural constraints.

2.2. *Introducing flexibility in XML query languages*

Querying XML documents with the standard query languages presented in the previous section allows the specification of structure related constraints (i.e. the expected path structure) that are matched against the document structure. However, an inaccurate path expression provided to an XQuery/XPath engine may lead to the common search issue named "too few or too many results"[16], where a query produces too many or too few results (or even none). For this reason in the last decade several approaches have been proposed aimed to introduce flexibility in the specification of the document structure by using various techniques among which Fuzzy Set Theory.

In 2006 Buche et al.[17] proposed a fuzzy-based XML querying approach able to perform approximate comparisons between the query and the data tree structures. This approach introduces flexibility into the query evaluation phase by creating a set of new queries, named *approximate queries*, by allowing the deletion, the renaming and the insertion of nodes in the original query q. Each approximate query is associated with a transformation cost. The approximate queries are evaluated and their results are aggregated to produce the *approximate result* of q. This technique supports the specification of imprecise data gathered from the Web via possibility distributions.

By another approach[18] a fuzzy set based extension of XQuery was proposed, which allows users to express flexible preferences on XML documents. The XQuery language is extended by this approach in order to allow user defined fuzzy constraints to be used in query expressions. However the approach does not introduce any specific flexible constraints on the target document structure. Also Thomson[19] proposed an approach that allows the user to define her own set of flexible constraints defined as fuzzy sets, to enhance XQuery in a business transactions environment. In all the approaches presented so far, the focus is on giving users the possibility to specify flexible constraints on the document content, but no novelty is introduced related to specifying flexible constraints on the document structure.

A set of approaches have been working around the idea of *path expression relaxation*. Schlieder[20] proposed the *approXQL* language that includes an algorithm for tree matching that first expands the user query by providing a set of similar queries obtained by node insertion, rename and delete operations, and then executes the new set instead of the original query (similarly to [17] but applied to a specific XML query language).

FlexPath[21] was the first proposal of an approximate matching of query constraints, who defined a formalization of the possible relaxations in the evaluation of the structure of the specified queries.

By another approach[22] the score computation of a retrieved XML fragment takes into account the number of edges that separate the fragment from the leaf node that actually contains the matched keywords.

Several other approaches have been proposed in the last years mostly applying tree-edit distances[23,24], or computing *semantic distances* among XML elements by using semantic networks (e.g., WordNet)[25].

With respect to the approaches presented in the literature, FleXy allows the user to specify flexible constraints on both content and structure of XML documents, conjugating the experiences de-

rived from both the Database and Information Retrieval research communities.

3. An Approach to introduce flexible constraints in XQuery-FT

In this section we present FleXy, an extension of XQuery-FT that introduces a set of flexible constraints to express structure related and content related constraints. FleXy was initially presented in[12] and formalized in[10,11].

The FleXy language offers a tolerance to vagueness and imprecision in XML retrieval by means of the following features:

- the possibility of retrieving only the most relevant parts of documents;
- a flexible approach to the approximate matching of structural query conditions;
- the use of vague predicates in query formulation to express flexible constraints on stored data (e.g., strings, numbers, dates);
- a tolerance to tag name misspelling.

The proposed approach allows to obtain a ranking of the retrieved fragments either based on the evaluation of the structure related constraints only, or the content related constraints only, or on a combination of them (which may also be decided by a user).

As an application of this flexible query language on XML documents, in section 4 we present PatentLight[13,14], a search application conceived to query huge collections of XML patents.

The flexible language FleXy includes the *structure related constraints* `below` and `near`, and the *content-based flexible constraint* `around`. Moreover the constraint `similar` has been defined on tag names; we will describe how the combination of the content-based and the structure-based evaluation of flexible constraints can improve the effectiveness of an XML retrieval system.

The new features of the language have been implemented on top of the open source engine BaseX [26].

3.1. Flexible Structure Related Constraints

This section describes the structure related constraints `below` and `near`, that have been defined as *XQuery-FT axes*; they have been formally defined with a syntax similar to the XQuery-FT axes `child` or `descendant`[10].

The evaluation functions of the new axes `near` and `below` compute the path relevance score of a document path with respect to the path specified in the query. Each score is a value in the interval $[0,1]$, where 1 represents an exact match between the document fragment path and the query path, while values less than 1 are assigned to target nodes far from the context node. Nodes with a path relevance score of zero will not be retrieved as they are not relevant to the user query.

The Constraint Below: The evaluation of the constraint `below` extends the XQuery `descendant` axis evaluation by computing, for each retrieved node, a path relevance score that is inversely proportional to the path distance from the ideal path.

For example, in the query `/Book/below::name` the node labeled `Book` is the *context* node for the `below` axis evaluation, while all nodes labeled `name` are the *target* nodes. The ideal path is the one where the target node is a direct child of the context node. Based on the example in Fig.1 the elements retrieved in decreasing order of relevance estimate are: (i) the node Book/name; and (ii) the nodes `Book/author/name`.

An important observation is that, although the XQuery-FT standard constraints could allow to formulate complex queries with a behavior similar to the one associated with the constraint `below`, the use of explicit flexible constraints is clearly more user-oriented.

Given a context node `c` and a target node `t`, the path relevance score for the `below` axis evaluation can be computed as follows (although of course alternative evaluation functions can be defined):

$$w_{below}(c,t) = \frac{1}{|desc\_arcs(c,t)|} \qquad (1)$$

Where `desc_arcs(c, t)` is a function that, given two XML nodes c and t (where t must be a descendant of node c), returns the set of descending arcs

from c to t. The score computed for the `below` axis is inversely proportional to the distance of the nodes c and t, thus giving to nodes closer to the context node a higher score than the one given to nodes far from the context node.

The Constraint Near: The `near` constraint allows to retrieve target nodes that are *in the neighbourhoods* of the context node, taking into account all possible directions (hence not only the descendant nodes, but also siblings and ancestors).

It is possible to specify a parameter within the `near` constraint to indicate the maximum distance allowed between the context node and the target node: nodes reachable with more than n arcs from the context node will be excluded from the retrieved elements. This parameter allows users to control the evaluation of the `near` constraint, and to restrict the search space.

Based on the previous considerations, for each matching node the `near` constraint evaluation computes a relevance score that is inversely proportional to the distance of the target node from the context node (following any direction in the document tree).

As an example, (see Fig.1) the evaluation of the query `Book/near(2)::editor` will consider the node labeled `Book` as the context node, while the `editor` node is the target node matched. The result is the entire fragment in Fig.1, since the node `collection` is the common node allowing to retrieve the information required in the query.

In this case the evaluation function that computes the path relevance score for the `near` axis is defined in Equation 2. Given the context node c, the target node t, l is the maximum allowed distance, and $arcs(c,e)$ is a function that returns the set of arcs in the shortest path between c and t.

$$w_{near}(c,t,l) = \begin{cases} \frac{1}{|arcs(c,t)|} & if |arcs(c,t)| \leqslant l \\ 0 & else \end{cases} \quad (2)$$

Like in the computation of the `below` score, this value is inversely proportional to the distance of the context node from the target node. The function assigns higher values if the target node is closer to the context node, and lower values as the distance increases.

On a performance point of view the evaluation of the FleXy axes introduces a noticeable overhead: a linear increasing evaluation timing is evident when a single huge document composes the queried collection, while a constant performance ratio is measured for collections composed of multiple small/medium XML documents.

3.2. Flexibility in Content and Tag Names Constraints

The flexible constraints `around` and `similar` can be applied to numbers/dates and tag names respectively.

The constraints evaluation for a given value (i.e., a leaf node content or an element tag name) produces a numeric score which expresses the satisfaction degree of the constraint when applied to that particular value.

The constraint "around" is used to specify a flexible condition in which the closeness between the content value specified in the query and the value stored in the document is evaluated. This flexible constraint relaxes the usual Boolean condition of exact matching to a precise value specified in the query. The constraint `around` is useful when the user wants to specify in a query an approximate numeric or date value, and not an exact value to be retrieved. As an example let us consider an XML file storing books with their number of pages. When a user is looking for a small book, she can quantify this concept saying that the number of pages has to be close to 100. In this example the query would be `Book/pages[text() around 100]`.

Regarding the semantics of the flexible constraint, it relies on definition of the membership function of the fuzzy subset associated with the concept *a*round n; this function expresses the compatibility between the stored numeric value and the concept linguistically express by the flexible constraint *a*round n. For integers and dates, the fuzzy membership function can be defined as a symmetric triangular function centered in $x \in X$.

The constraint "similar" allows to retrieve document nodes with a tag name typographically similar to the tag name specified in the query. This

means that the tag name is evaluated with a tolerance for characters mismatching. This is useful, because often XML tags can be introduced with typos. Consider as an example the tag representing the editor of a book in Fig.1. If a user searches for books whose editor is Bompiani, by using the `similar` constraint also a node named *editor-name* would be retrieved. The query should be in this case specified as follows: `Collection/similar(editor-name)[text()=''Bompiani'']`. In order to assess string similarity, the evaluation of the constraint `similar` relies on the Levenshtein distance[29].

3.3. Evaluation of queries with multiple flexible constraints

An important issue concerns the evaluation of queries that involve multiple flexible constraints in fragment selection.

For each considered document fragment, the evaluation of a flexible constraint associates a relevance score in the range $[0, 1]$ with each node involved in the flexible part of the query.

While in flat queries this can be done in a simple way, a particular observation should be made for those complex queries where the selection node that needs to be ranked appears in a different tree branch than the one/ones that use the flexible constraints, thus obtaining a path relevance score.

In this case, for each retrieved fragment we may have different scores associated with the node to be retrieved, depending on different paths that in the same document satisfy the flexible constraint; therefore how the final score must be computed and assigned to the result is an important issue. A reasonable solution to address this situation is to apply the `max` aggregation operator. Although this choice is quite natural to obtain an optimistic aggregation, other aggregation schemes could be investigated.

4. An application of Flexy on Patent Search Retrieval

In this section we present an application of FleXy to a particular domain, XML Patent Retrieval, which may greatly benefit of the expressive power given by this language.

Patent Information Retrieval (PIR) is an important specialized branch of Information Retrieval, which is aimed to support users, often professionals such as patent attorneys, in retrieving patents that satisfy their information needs[30].

Today patents are publicly available through collections such as USPTO (United States Patent and Trade Office), EPO (European Patent Office) and WIPO (World Intellectual Property Organization), which are also encoded in XML since 2012.

Each collection contains several thousands of patents, and it grows up year by year; therefore the cost of retrieving the state of the art (in this domain called *prior-art*) for filing patents or defending a claim of infringement purposes is increasing with time.

For the above reasons Patent Retrieval stimulates a high interest in the scientific community, and it is also considered a complex challenging task because the employed vocabulary is composed of several specialized or technical words. Moreover, often the obfuscation of content is intentional by writers who wish their patents difficult to retrieve.

Most approaches presented in the literature, based on *keyword extraction* or *query expansion* techniques, proved to produce poor results[14].

As shown in [13,14], the FleXy language can produce good results with respect to both recall and precision in the patent retrieval domain.

4.1. How the flexible constraints works in patent search

This section explains how the flexible constraints introduced in the previous section are used to exploit the patent search task. In particular, the proposed approach has been defined and tested in[31] on the USPTO patent collection that can be freely downloaded on the Web. USPTO is the corpus adopted by most patent search applications such as Google Patents and PatentsSearcher. The first prototype was improved in[13] by introducing the possibility to retrieve fragments with different tag names w.r.t. those expressed in the query. This feature is useful when we query collections of which we roughly know the internal structure (tag names and node positions), or

when we want to apply the same query to a composition of patent collections that contain more or less the same information stored in different tag nodes (for instance the node `Orgname` instead of the node `Last-Name`). See Fig.2.

Indeed it was outlined[13] that EPO and WIPO patent documents show more or less the same structure of USPTO, even if with different tags. For this reason the `similar` constraint was used to perform a set of experiments on a collection composed by both USPTO and WIPO documents.

The proposed approach allows users to search patents by means of a keyword based query, thus completely hiding to users the complexity of the syntax of XQuery-FT.

4.1.1. *Keyword-based Query: the approach*

An important functionality of the flexible patent search approach[31] is to categorize patents by exploiting their XML structure. In this way the engine organizes the XML patents into meaningful semantic XML elements covering the main patent information. The categorization process described below can easily capture what the user topical search intent is by identifying the possible interpretations associated with a patent. By analyzing the patents in the USPTO collection, four main semantic categories were identified[31]: *People* containing details about inventors and other people involved, *Title* of the patent, *Description* with a full textual description of the invention, and *Claims* containing a summary of the main characteristics and purposes of the invention. An analysis of the other collections showed that the same categorization can be applied also to them: the structure of patent documents appear similar in all collections under analysis.

The above categories have been identified by the XML elements: *People* (the associated elements are `Applicants`, `Agents`, `Assignee`, `Examiners` as shown in the USPTO patent fragment depicted in Fig.2), *Title* (`title`), *Description* (`description`), and *Claims* (`claims`).

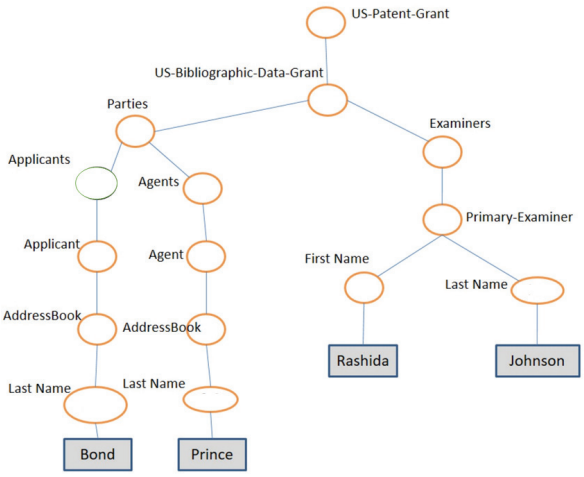

Fig. 2. An XML document fragment from an USPTO patent.

A user specified keyword based query (here below "`query terms`") is automatically rewritten into four distinct flexible queries, one for each of the four categories. The structure of each query is predefined in order to search the query terms in pre-established elements. Below an example for the category *People*, for further details see previous works [31,13].

```
People:
applicants/near::Last-Name[
text() contains text "query terms"]
```

In case of user queries requiring a *name* of a person, it is supposed that he/she is interested in finding inventors of patents. However, it is important to notice that by the approach also patents containing the *name* with a different role will be retrieved.

In the engine, if the user specifies the `similar` constraint the set of flexible queries would change accordingly, as shown below (again we show only the category *People*, further details are available in[13]):

```
People:
similar(applicants)/near::Last-Name[
text() contains text "query terms"]
applicants/near::similar(Last-Name)[
text() contains text "query terms"]
```

Note that the `similar` constraint is applied to each node of the query path (in the example `applicants` and `Last-Name`), and therefore each category will contain two queries instead of one.

The retrieved fragments are ranked according to the combination of two values: the degree of structural relevance based on the evaluation of the flexible constraints, and the degree of relevance obtained by the full-text scoring of the XQuery Full Text language (the prototype in[31] uses the BaseX system[26]). The approach privileges the structural ranking w.r.t. the content based relevance since it was observed that the paragraphs most related to the invention are usually structurally closer to the tag `Description`. For details see[14,13].

4.2. The PatentLight tool

The PatentLight tool has been developed on top of the BaseX system[26], inheriting its indexing system and query execution engine. During the querying process each class query is executed independently; one of the main characteristics of the approach is that each query produces four distinct ranked lists, one for each class (*People, Title, Descriptions, Claims*).

To the aim of exploring the usefulness of FleXPath on the patent collection several different searches were performed, fully presented in[13], to the aim of comparing PatentLight and Google Patents w.r.t the prior-art retrieval task. In this task users really need to find the highest number of relevant patents as possible; in the preliminary evaluations the first 50 results for each search were carefully checked, and they were compared with the results provided by Google Patents for the same query. The tests showed that in most cases PatentLight retrieved the same number or a higher number of relevant patents w.r.t. Google Patents within the first 50 results. Moreover the presentation of the results in four distinct lists may be very useful to easily find the relevant documents.

5. Conclusions and Future Work

In this paper we have reviewed some fuzzy approaches to flexible querying in XML. In particular the FleXy language has been synthesized as well as its application to the patent retrieval domain. With the possibility to express vagueness in structural constraints provided by FleXy, besides the possibility of formulating keyword-based search as addressed by the XQuery Full-Text extension, the user can formulate queries that explicitly represent her/his needs, in a way tolerant to imprecision, thus having a full control on the desired evaluation process.

The application of FleXy to the patent search domain has shown to be particularly promising in the case in which we need to query several collections that do not share the same document structure or tag vocabulary. At the present time, the direction in which the main patent organizations (EPO, USPTO, WIPO) appear to be moving is finding a common standard representation for XML patents, but the situation could become more challenging as soon as the collections of other organizations (e.g. the Japan Patent Office, the Eurasian Patent Organization, etc..) will be indexed by patent search engines.

Other important applications of a flexible query language for XML documents can be found in the retrieval of scientific articles, or to inquire data collections coming from the *Internet of Things*; XML has been widely studied as data exchange format in this context[32].

References

1. G.Pasi, G. Bordogna, *The Role of Fuzzy Sets in Information Retrieval*, In "On Fuzziness", vol. 299, Studies in Fuzziness and Soft Computing, ed. R. Seising, E. Trillas, C. Moraga,and S. Termini, 2003, pages 525–532.
2. G. Pasi, *Fuzzy Sets in Information Retrieval: State of the Art and Research Trends*, In Fuzzy Sets and Their Extensions: Representation, Aggregation and Models, Studies in Fuzziness and Soft Computing, ed. H. Bustince, F. Herrera, and J. Montero, vol. 220, 2008, pages 517–535.
3. http://www.w3.org/XML/.
4. S. Amer-Yahia, N. Koudas, A. Marian, D. Srivastava, D. Toman: *Structure and Content Scoring for XML.* In: VLDB 2005, pp. 361–372 (2005).
5. C. Yu, H. V. Jagadish: *Querying Complex Structured Databases*. In: VLDB 2007, September 23-27 2007, University of Vienna, Austria, pp. 1010–1021.
6. *W3C XML Requirements* http://www.w3.org/.
7. *W3C XPath Requirements* http://www.w3.org/TR/xpath/.

8. *W3C XQuery Requirements* http://www.w3.org/TR/xquery/.
9. *W3C XQuery and XPath Full Text 1.0 Requirements* http://www.w3.org/TR/xpath-full-text-10/.
10. E. Panzeri, G. Pasi, *An Approach to Define Flexible Structural Constraints in XQuery*, In Active Media Technology, Lecture Notes in Computer Science, ed. R. Huang, A. Ghorbani, G. Pasi, T. Yamaguchi, N. Yen,and B. Jin, Springer Berlin Heidelberg, 2012, pages 307–317.
11. E. Panzeri and G. Pasi, *Flex-BaseX: an XML engine with a flexible extension of XQuery full-text*, In the 36th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '13, Dublin, Ireland - July 28 - August 01, 2013, pages 1083–1084.
12. E. Damiani, S. Marrara, and G. Pasi, *A flexible extension of XPath to improve XML querying*, Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2008, Singapore, July 20-24, 2008, pages 849–850.
13. S. Marrara, G. Pasi, *A Flexible Tool for Cross-Collection Patent Search*, IIR 2015, May 25–26 2015, Cagliari, Italy.
14. S. Marrara, G. Pasi, *Flexibility in Patent Search* IFSA-EUSFLAT 2015, June 30th – July 3rd, Gijn, Asturias (Spain).
15. A. Trotman, B. Sigurbj´ornsson, *Narrowed Extended XPath I (NEXI)*. In INEX 2004. LNCS, vol. 3493, pp. 16–40. Springer, Heidelberg (2005).
16. C. D. Manning, P. Raghavan, and H. Sch´utze, *Introduction to Information Retrieval*, 1st Ed Cambridge University Press, 2008.
17. P. Buche, J. Dibie-Barthlemy, and F. Wattez, *Approximate querying of XML fuzzy data*, in Proceedings of the 7th International Conference FQAS, Milano, Italy, 2006.
18. M. Goncalves, and L. Tineo, *FuzzyXQuery*, In Soft Computing in XML Data Management, Studies in Fuzziness and Soft Computing, Springer Berlin Heidelberg, 2010, vol. 255, pp 133–163.
19. EJ Thomson Fredrick, and G. Radhamani. *Fuzzy logic based XQuery operations for native XML database systems.* International Journal of Database Theory and Application 2.3 (2009), pages 13–20.
20. T. Schlieder, *Schema-Driven Evaluation of Approximate Tree-Pattern Queries*. In Advances in Database Technology - EDBT2002, vol. 2287, Lecture Notes in Computer Science, 2002, pages 514–532.
21. S. Amer-Yahia, L. V. S. Lakshmanan, and S. Pandit. *FleXPath: Flexible Structure and Full-Text Querying for XML*, In Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, SIGMOD '04, pages 83–94, 2004.
22. K. Sauvagnat, L. Hlaoua, and M. Boughanem. *XFIRM at INEX 2005: Ad-Hoc and Relevance Feedback Tracks*. In Advances in XML Information Retrieval and Evaluation, vol. 3977, Lecture Notes in Computer Science, 2005, pages 88–103.
23. A. Niermann and H. V. Jagadish, *Evaluating Structural Similarity in XML Documents*, in Proceedings of the Fifth International Workshop on the Web and Databases, WebDB '02, Madison, Wisconsin, Usa, pages 61–66, 2002.
24. G. Buratti and D. Montesi, *Ranking for Approximated XQuery Full-Text Queries*, In Sharing Data, Information and Knowledge. Vol 5071. Lecture Notes in Computer Science, 2008, pages 165–176.
25. J. Tekli and R. Chbeir, *A Novel XML structure comparison framework based on sub-tree commonalities and label semantics*. In Web Semantics 11, March 2012, pages 14–40.
26. E. Panzeri and G. Pasi,*Flex-BaseX: an XML engine with a flexible extension of XQuery Full-Text*, In the 36th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '13, Dublin, Ireland, July 28 – August 01, 2013.
27. XML Schema http://www.w3.org/XML/Schema.
28. E. Damiani, N. Lavarini,B. Oliboni,and L. Tanca, *An approximate query environment for XML data*. In Fuzzy logic and the internet, Studies in Fuzziness and Soft Computing (Vol. 137, pp. 71–94, 2004).
29. V. I. Levenshtein, *Binary codes capable of correcting deletions, insertions and reversals*. In Soviet Physics-Doklady, 10, 707–710, 1996.
30. M. Lupu, A. Hanbury. *Patent Retrieval Foundations and Trends in Information Retrieval*. vol 7(1), pages 1–97, 2013.
31. S. Calegari, E. Panzeri, G. Pasi, *PatentLight: a Patent Search Application*, Proceedings of lliX 2012, Nijmegen, The Netherlands.
32. D. Uckelmann, M. Harrison, F. Michahelles, *An Architectural Approach Towards the Future Internet of Things*, In Architecting the Internet of Things, Springer Berlin Heidelberg, pages 1–24, 2011.