XML Retrieval with a Natural Language Interface

Xavier Tannier[1] and Shlomo Geva[2]

[1] École Nationale Supérieure des Mines de Saint-Etienne, 158 Cours Fauriel, F-42023 Saint-Etienne, France
tannier@emse.fr

[2] Centre for Information Technology Innovation, Faculty of Information Technology, Queensland University of Technology, GPO Box 2434, Brisbane Q 4001, Australia
s.geva@qut.edu.au

Abstract. Effective information retrieval in XML documents requires the user to have good knowledge of document structure and of some formal query language. XML query languages like XPath and XQuery are too complex to be considered for use by end users. We present an approach to XML query processing that supports the specification of both textual and structural constraints in natural language. We implemented a system that supports the evaluation of both formal XPath-like queries and natural language XML queries. We present comparative test results that were performed with the INEX 2004 topics and XML collection. Our results quantify the trade-off in performance of natural language XML queries vs formal queries with favourable results.

1 Introduction and Motivation

Applications of Natural Language Processing to Information Retrieval have been extensively studied in the case of textual (flat) collections (see overviews [1, 2, 3, 4]). Among other techniques, linguistic analysis of queries was meant to bring about decisive improvements in retrieval processes and in ergonomy. However, only few linguistic methods, such as phrasal term extraction or some kinds of query expansion, are now commonly used in information retrieval systems.

The rapidly growing spread of XML document collections brings new motivating factors to the use of natural language techniques:

– Benefits that can be gained from the use of natural language queries are probably much higher in XML retrieval than in traditional IR. In the later, a query is generally a list of keywords which is quite easy to write. In XML retrieval, such a list is not sufficient to specify queries on both content and structure; for this reason, advanced structured query languages have been devised.

  XML is now widely used, particularly on the Internet, and that implies that novice and casual users ought to be able to query any XML corpus. From that perspective, two major difficulties arise, because we cannot expect such users to:

  - learn a complex structured formal query language (a language with formalized semantics and grammar, as opposed to natural language);
  - have full knowledge of the DTD and its semantics.
- In structured documents, a well-thought and semantically strong structure formally marks up the meaning of the text; this can make easier query "understanding", at least when this query refers (partly) to the structure.
- Finally, formal queries do not permit information retrival in heteregenous collections (with different and unknown DTDs). A natural language interface could resolve this problem, since users can express their information need conceptually.

Note that these comments could be made about the database domain too, and that the issues seem quite similar. Many natural language interfaces for databases have been developed, most of them transforming natural language into Structured Query Language (SQL) (see [5, 6, 7] for overviews). But the problem is different for the following reasons:

- Unlike databases, XML format looks set to be used and accessed by the general public, notably through the Internet. Although unambiguous, machine-readable, structured and formal query languages are necessary to support the retrieval process (in order to actually extract the answers), the need for simpler interfaces will become more and more important in the future.
- Database querying is a strict interrogation; it is different to Information Retrieval. The user knows what kind of information is contained in the database, her information need is precise, and the result she gets is either right or wrong. This means that the natural language analysis must interpret the query perfectly and unambiguously, failing which the final answer is incorrect and the user disatisfied.

  In XML IR, as well as in traditional IR, the information need is loosely defined and often there is no perfect answer to a query. A natural language interface is a part of the retrieval process, and thus it can interpret some queries imperfectly, and still return useful results. The problem is then made "easier" to solve...

2 INEX, NLPX Track and NEXI

2.1 INEX

The Initiative for Evaluation of XML Retrieval, INEX [8], provides a test collection consisting of over 500 Mbytes of XML documents, topics and relevance assessments. The document set is made up of 12,107 articles of the IEEE Computer Society's publications. Topics are divided into two categories:

- *Content-and-Structure* (CAS) queries, which contain structural constraints.
  e.g.: *Find* paragraphs *or* figure-captions *containing the definition of Godel, Lukasiewicz or other fuzzy-logic implications.* (Topic 127)

– *Content-Only* (CO) queries that ignore the document structure.
  e.g.: Any type of coding algorithm for text and index compression. (Topic 162)

This article focuses on CAS topics. Figure 1 shows an example of CAS topic. The `description` element is a natural language (English) description of the user's information need; The `title` is a faithful translation of this need into a formal XPath-like language called Narrowed Extended XPath I (NEXI) [9]. The `narrative` part is a more detailed explanation of the information need.

```
<inex_topic topic_id="130" query_type="CAS">

    <title>
        //article[about(.//p,object
        database)]//p[about(.,version management)]
    </title>
    <description>
        We are searching paragraphs dealing with version management
        in articles containing a paragraph about object databases.
    </description>
    <narrative>
        The elements to be considered relevant are ...
    </narrative>
    <keywords>object database version management</keywords>

</inex_topic>
```

Fig. 1. An example of CAS topic

The participants in the *ad-hoc* INEX task use only NEXI titles in order to retrieve relevant elements. We adapted our system so that it takes the topic description (natural language expression depicting the same query) as input, and returns a well-formed NEXI title. Going through this pivot language presents many advantages: it allows the use of an existing NEXI search engine in the retrieval process. Furthermore a user can still specify her query in this formal language if she prefers to. Finally, we can evaluate the translation by comparing the translated queries with the original hand-crafted NEXI titles. On the other hand, the transformation to a pre-existing restrictive language may result in loss of information.

2.2 NEXI

NEXI CAS queries have the form **//A[B]//C[D]** where A and C are paths and B and D are filters. We can read this query as *"Return C descendants of A where A is about B and C is about D"*. B and D correspond to disjunctions or conjunctions of 'about' clauses **about(//E, F)**, where E is a path and F a list of terms. The `'title'` part of Fig. 1 gives a good example of a query formulated in NEXI. More information about NEXI can be found in [9].

3 Description of Our Approach

Requests are analysed through several steps:

1. A part-of-speech (POS) tagging is performed on the query. Each word is labeled by its word class (*e.g.:* noun, verb, adjective...).
2. A POS-dependant semantic representation is attributed to each word. For example the noun *'information'* will be represented by the predicate *information(x)*, or the verb *'identify'* by *evt*(e_1*, identify)*.
3. Context-free syntactic rules describe the most current grammatical constructions in queries and questions. Low-level semantic actions are combined with each syntactic rule. Two examples of such operations, applied to the description of topic 130 (Fig. 1), are given in Fig. 2. The final result is a logical

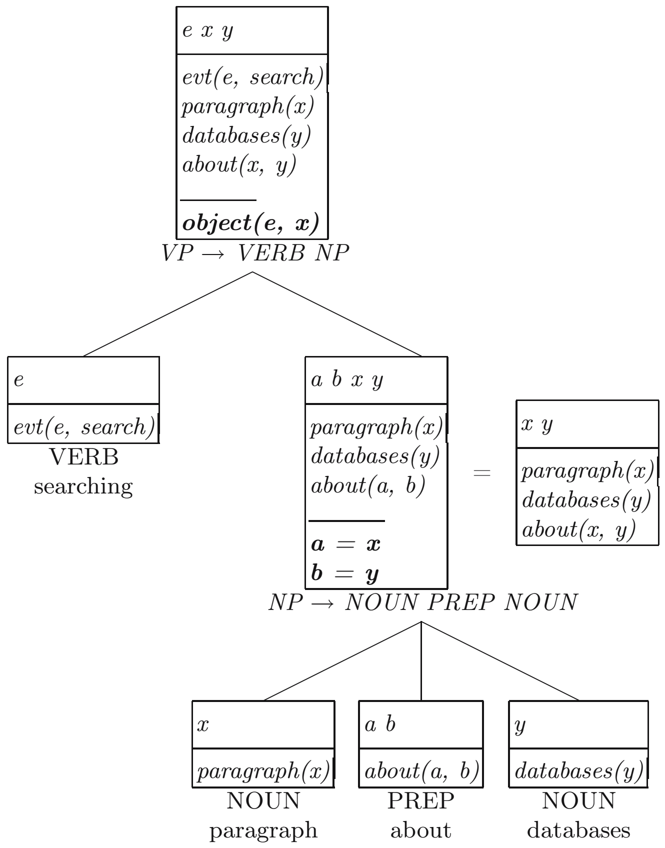


Fig. 2. Example of rule application for the verbal phrase "*searching paragraphs about databases*" (rules *NP → NOUN PREP NOUN* and *VP → VERB NP*). Basic semantic representations are attributed to part-of-speeches (leaf components). When applying syntactic rules, components are merged and semantic actions are added (here identity relations and verbal relation predicate – bold predicates).

representation shown in the left part of Fig. 3. This representation is totally independant from the queried corpus, it is obtained by general linguistic operations.
4. The semantic representation is then reduced with the help of specific rules:
   - a recognition of some typical constructions of a query (*e.g.: Retrieve + object*) or of the corpus (*e.g.: "an article written by [. . .]"* refers to the tag *au – author*);
   - and a distinction between semantic elements mapping on the structure and, respectively, mapping on the content;

   This part is the only one that uses corpus-specific information, among which the DTD, a dictionary of specific tag name synonyms (*e.g.*: *paper*=`article`), some simple ontologic structures (*"a article citing somebody"* refers to bibliography in INEX collection). Figure 3 shows the specific rules that apply to the example.
5. A treatment of relations existing between different elements;
6. The construction of a well-formed NEXI query.

Steps 1 to 5 are explained in more details in [10], as well as necessary corpus knowledge and the effect of topic complexity on the analysis.

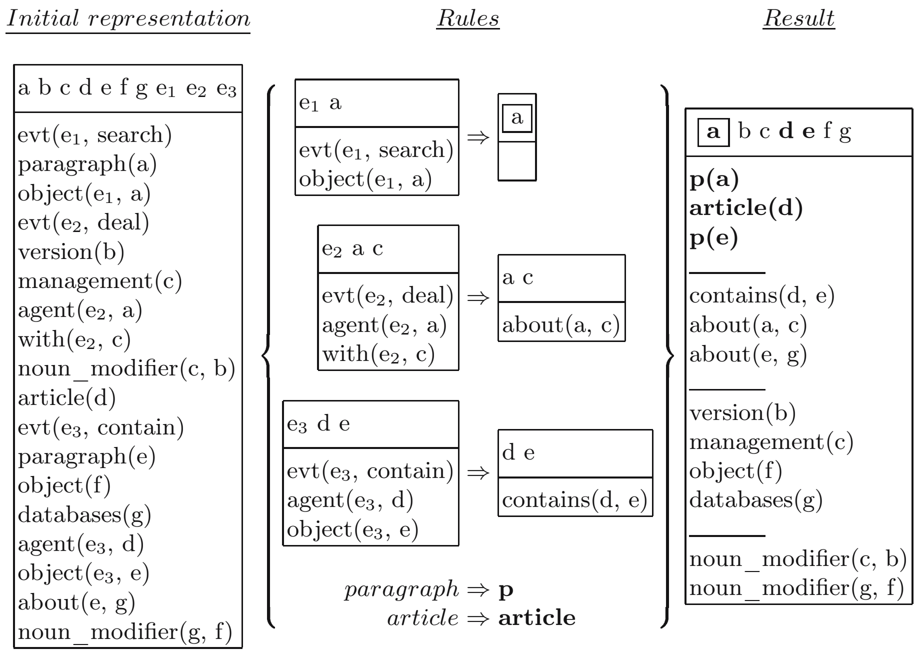


Fig. 3. The semantic analysis of topic 130 (left), is reduced by some generic rules (center), leading to a new representation (right). Bold predicates emphasize words representing XML tag names and the framed letter stands for the element that should be returned to the user. The first three rules deal with verbal phrases *"to search sth"*, *"to deal with sth"* and *"to contain sth"*.

The representation obtained at the end of step 5 does not depend on any retrieval system or query language. It could be transformed (with more or less information loss) into any existing formal language.

3.1 Getting to NEXI

Transformation process from our representation to NEXI is not straightforward. Remember that a NEXI query has the form //**A[B]**//**C[D]**.

- At structural level, a set of several tag identifiers (that can be DTD tag names or wildcards) has to be distributed into parts A, B, C and D, that we respectively call support requests, support elements, return requests and return elements.
- At content level, linguistic features (like `noun_modifier` in the example) cannot be kept and must be transformed in an appropriate manner.

Structural Level. These four parts A, B, C and D are built from our representation (Fig. 3) in the following way:

- C is the ’framed’ (selected) element name (see Fig. 3 and its caption);
- D is composed of all C children (relation *contains*) and their textual content (relation *about*);
- A is the highest element name in the DTD tree, that is not C or one of its children;
- B is composed of all other elements and their textual content.

Wildcard-identified tags of the same part are merged and are considered to be the same element. See an example in Sect. 4.

Table 1. Examples of linguistic features and of their NEXI equivalents

Predicate	Initial text	Representation	NEXI content

`noun_property`	“definition of a theorem”	*definition(a)*
		theorem(b)
		noun_property(a, b, of)
			“defition of theorem”

`noun_modifier`	“version management”	*version(a)*
		management(b)
		noun_modifier(b,)
			“version management”

`adjective`	“digital library”	*digital(a)*
		library(b)
		adjective(b, a)
			“digital library”

`disjunction`	“definition of Godel or Lukasiewicz”	*definition(a)*
		noun_property(a, b, of)
		$b = c \vee d$
		Godel(c)
		Lukasiewicz(d)
			“definition of Godel definition of Lukasiewicz”

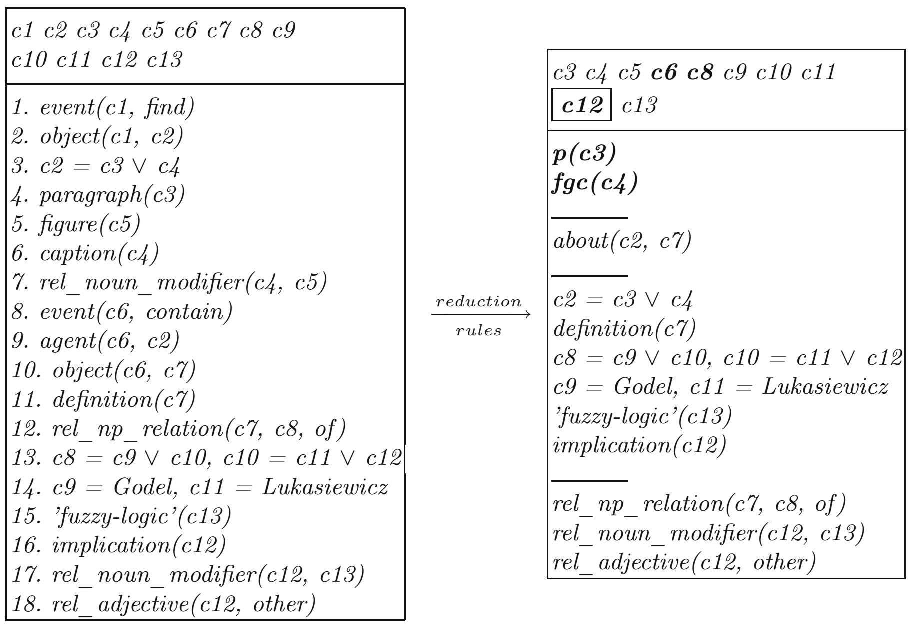
Content Level. The main linguistic predicates generated by our system are `np_property`, `noun_modifier`, `adjective` and disjunction or conjunction relations. NEXI format requires 'about' clauses to contain only textual content. In most cases we chose to reflect as far as possible the initial text, because the search engine can deal with noun phrases. In the case of disjunctions and conjunctions, for the same reason, we built separated noun phrases. Examples of each operation are given in Tab. 1.

The transformation of the semantic representation of Fig. 3 results in:

```
//article[(about(.//p, object databases))]//p[(about(.,
                version management))]
```

4 Example

We give here a significant example, with the analysis of topic 127 (INEX 2004). Several syntactic parsings could be possible for the same sentence. In practice a "score" is attributed to each rule release, depending on several parameters (among which distance between words that are linked, length of phrases, type of relations... Unfortunately we lack space to explain more precisely this process). In our sample topic only the best scored result is given.



c1 c2 c3 c4 c5 c6 c7 c8 c9
c10 c11 c12 c13

1. event(c1, find)
2. object(c1, c2)
3. c2 = c3 ∨ c4
4. paragraph(c3)
5. figure(c5)
6. caption(c4)
7. rel_noun_modifier(c4, c5)
8. event(c6, contain)
9. agent(c6, c2)
10. object(c6, c7)
11. definition(c7)
12. rel_np_relation(c7, c8, of)
13. c8 = c9 ∨ c10, c10 = c11 ∨ c12
14. c9 = Godel, c11 = Lukasiewicz
15. 'fuzzy-logic'(c13)
16. implication(c12)
17. rel_noun_modifier(c12, c13)
18. rel_adjective(c12, other)

$\xrightarrow[rules]{reduction}$

c3 c4 c5 ***c6 c8*** *c9 c10 c11*
$\boxed{\textbf{c12}}$ *c13*

p(c3)
fgc(c4)

about(c2, c7)

c2 = c3 ∨ c4
definition(c7)
c8 = c9 ∨ c10, c10 = c11 ∨ c12
c9 = Godel, c11 = Lukasiewicz
'fuzzy-logic'(c13)
implication(c12)

rel_np_relation(c7, c8, of)
rel_noun_modifier(c12, c13)
rel_adjective(c12, other)

```
//article//(p|fgc)[(about(., "definition of Godel" "definition of
    Lukasiewicz" "definition of fuzzy-logic implications"))]
```

Fig. 4. Semantic representations of topic 127, and automatic conversion into NEXI

(127) Find paragraphs or figure captions containing the definition of Godel, Lukasiewicz or other fuzzy-logic implications.

Figure 4 shows the three major steps of the analysis of topic 127. The left frame represents the result of step 3 (see Sect. 3). Some IR- and corpus-specific reduction rules are then applied and lead to right frame: terms *paragraph* and *figure-captions* are recognized as tag names **p** and **fgc** (lines 4 to 7); the construction *"c2 contains c7"* is changed into **about(c2, c7)** (lines 8 to 11). The other relations are kept.

Translation into NEXI is performed as explained above, disjunctions *c8* and *c10* result in the repetition of the term "*definition*" with preposition "*of*" and three distinct terms.

5 Processing NEXI Queries

5.1 XML File Inversion

In our scheme each term in an XML document is identified by three elements: its filename, its absolute XPath context, and its ordinal position within the XPath context. An inverted list for a given term is depicted in Tab. 2.

Table 2. Inverted file

Document	XPath	Position
e1303.xml	article[1]/bdy[1]/sec[6]/p[6]	23
e1303.xml	article[1]/bdy[1]/sec[7]/p[1]	12
e2404.xml	article[1]/bdy[1]/sec[2]/p[1]/ref[1]	1
f4576.xml	article[1]/bm[1]/bib[1]/bibl[1]/bb[13]/pp[1]	3
f4576.xml	article[1]/bm[1]/bib[1]/bibl[1]/bb[14]/pp[1]	2
g5742.xml	article[1]/fm[1]/abs[1]	7

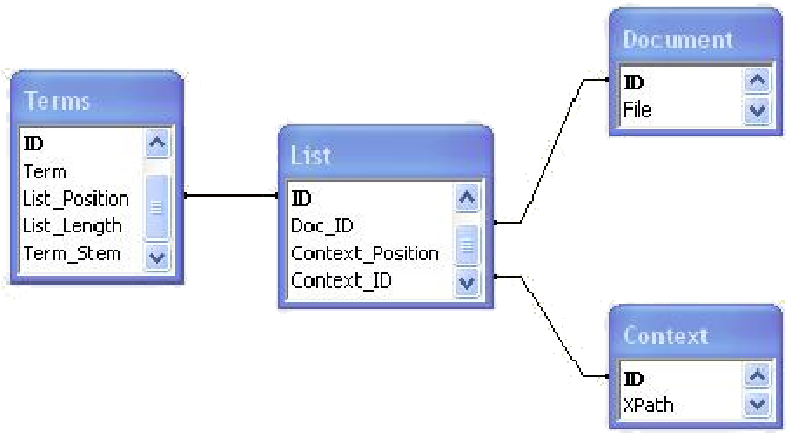


Fig. 5. Schema for XML Inverted File

In principle at least, a single table can hold the entire cross reference list (our inverted file). Suitable indexing of terms can support fast retrieval of term inverted lists. However, it is evident that there is extreme redundancy in the specification of partial absolute XPath expressions (substrings). There is also extreme redundancy in full absolute XPath expressions where multiple terms in the same document share the same leaf context (e.g. all terms in a paragraph). Furthermore, many XPath leaf contexts exist in almost every document (e.g. /article[1]/fm[1]/abs[1] in INEX collection). For these reasons we chose to normalize the inverted list table to reduce redundancy.

The structure of the database used to store the inverted lists is depicted in Fig. 5. It consists of four tables. The `Terms` table is the starting point of a query on a given term. The `Term_Stem` column holds the Porter stem of the original term. The `List_Position` is a foreign key from the `Terms` table into the `List` Table. It identifies the starting position in the inverted list for the corresponding term. The `List_Length` is the number of list entries corresponding to that term. The `List` table is (transparently) sorted by `Term` so that the inverted list for any given term is contiguous.

5.2 Ranking Scheme

Elements are ranked according to a relevance judgment score. Leaf and branch elements need to be treated differently. Data usually occur at leaf elements, and thus, our inverted list mostly stores information about leaf elements. A leaf element is considered candidate for retrieval if it contains at least one query term. A branch node is candidate if it contains a relevant child element. Once an element (either leaf or branch) is deemed to be a candidate for retrieval its relevancy judgment score is calculated. A heuristically derived formula (Equation (1)) is used to calculate the relevance judgment score of leaf elements. The same equation is used for both return and support elements. The score is determined from query terms contained in the element. It penalises elements with frequently occurring query terms (frequent in the collection), and it rewards elements with evenly distributed query term frequencies within the elements.

$$L = K^{n-1} \sum_{i=1}^{n} \frac{t_i}{f_i} \tag{1}$$

Here n is the number of unique query terms contained within the leaf element, K is a small integer (we used $K = 5$). The term K^{n-1} scales up the score of elements having multiple distinct query terms. We experimented with $K = 3$ to 10 with little difference in results. The sum is over all terms where t_i is the frequency of the i^{th} query term in the leaf element and f_i is the frequency of the i^{th} query term in the collection. This sum rewards the repeat occurrence of query terms, but uncommon terms contribute more than common terms.

Once the relevance judgment scores of leaf elements have been calculated, they can be used to calculate the relevance judgment score of branch elements. A naïve solution would be to just sum the relevance judgment score of each branch relevant children. However, this would ultimately result in root elements accumulating at the top of the ranked list, a scenario that offers no advantage over document-level retrieval. Therefore, the relevance judgment score of children elements should be somehow decreased while being propagated up the XML tree.

A heuristically derived formula (Equation (2)) is used to calculate the scores of intermediate branch elements:

$$R = D(n) \sum_{i=1}^{n} L_i \tag{2}$$

Where:

- n = the number of children elements
- $D(n) = 0.49$ if $n = 1$
  0.99 otherwise
- L_i = the i^{th} return child element

The value of the decay factor D depends on the number of relevant children that the branch has. If the branch has one relevant child then the decay constant is 0.49. A branch with only one relevant child will be ranked lower than its child. If the branch has multiple relevant children the decay factor is 0.99. A branch with many relevant children will be ranked higher than its descendants. Thus, a section with a single relevant paragraph would be judged less relevant than the paragraph itself, but a section with several relevant paragraphs will be ranked higher than any of the paragraphs.

Having computed scores for all result and support elements, the scores of support elements are added to the scores of the corresponding result elements that they support. For instance, consider the query:

//A[about(.//B,C)]//X[about(.//Y,Z)]

The score of a support element //A//B will be added to all result elements //A//X//Y where the element A is the ancestor of both X and Y.

Finally, structural constraints are only loosely interpreted. Elements are collected regardless of the structural stipulations of the topic. Ancestors or descendants of Y may be returned, depending on their score and final rank.

More information about this system can be found in [11].

6 Results

We tested our system using the INEX 2004 collection (set of topics and evaluation metrics). Recall/precision graphs have been calculated by the official INEX evaluation program. In the following we call S_{adhoc} the system using official, hand-crafted NEXI titles. S_{NLP} is the same system, but using natural language queries, automatically translated into NEXI.

S_{adhoc} is ranked 1^{st} from 52 submitted runs in the task, with an average precision of 0.13. S_{NLP} is ranked 5^{th} with an average precision of 0.10, and 1^{st} among the systems using natural language queries.

The Recall/Precision curves are presented in Fig. 6. The top bold dashed curve represents results for S_{adhoc}, the lower bold one is S_{NLP} curve, and the other curves are all the official runs submitted at INEX 2004.

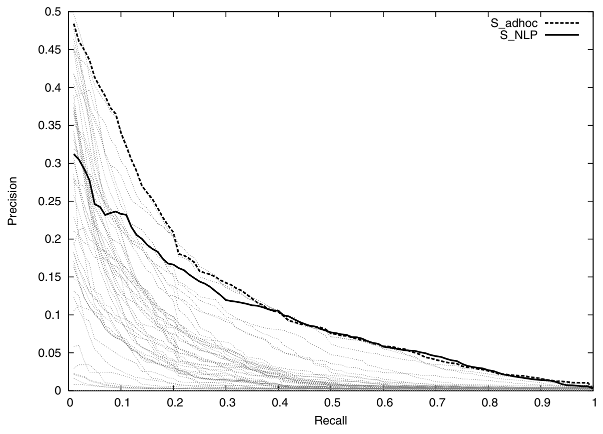


Fig. 6. INEX'04 VCAS Recall/Precision curve averaged over all topics and all metrics

The precision loss caused by the natural language interface is limited. S_{NLP} looses only four ranks with this interface, and still outperforms most of ad-hoc systems. We think that this trade-off is very interesting; indeed, as we pointed out in the introduction, the benefits brought by a natural language interface compensate for the precision loss, at least for non-expert users[1].

7 Conclusion

In this paper we presented an XML retrieval system that allows the user to express a query over an XML collection, using both structural and content retrieval cues, in natural English expression. An NLP module analyses this expression syntactically and semantically, applies some specific rules and translates the result into a formal query language. This intermediate language is then processed by a backend system.

This system had been tested with INEX 2004 collection, topics, and relevance assessments and with good results. This study shows that natural language queries over XML collections can offer promising prospects for deploying in general public applications.

[1] We can note that before an online NEXI parser became available for INEX topic developers, the majority of submitted topics were not well formed (depicting the wrong meaning) and/or syntactically incorrect. However INEX participants are XML retrieval professionals that have at least a good knowledge of XPath and NEXI. The task would have been much more difficult for casual users.

References

[1] Smeaton, A.F.: Information Retrieval: Still Butting Heads with Natural Language Processing? In Pazienza, M., ed.: Information Extraction – A Multidisciplinary Approach to an Emerging Information Technology. Volume 1299 of Lecture Notes in Computer Science. Springer-Verlag (1997) 115–138
[2] Smeaton, A.F.: Using NLP or NLP Resources for Information Retrieval Tasks. [12] 99–111
[3] Arampatzis, A., van der Weide, T., Koster, C., van Bommel, P.: Linguistically-motivated Information Retrieval. In Kent, A., ed.: Encyclopedia of Library and Information Science. Volume 69. Marcel Dekker, Inc., New York, Basel (2000) 201–222
[4] Sparck Jones, K.: What is the role of NLP in text retrieval? [12] 1–24
[5] Androutsopoulos, I., G.D.Ritchie, P.Thanisch: Natural Language Interfaces to Databases – An Introduction. Journal of Natural Language Engineering **1** (1995) 29–81
[6] A.Copestake, Jones, K.S.: Natural Language Interfaces to Databases. The Knowledge Engineering Review **5** (1990) 225–249
[7] Perrault, C., Grosz, B.: Natural Language Interfaces. Exploring Articial Intelligence (1988) 133–172
[8] Fuhr, N., Lalmas, M., Malik, S., Szlàvik, Z., eds.: Advances in XML Information Retrieval. Third Workshop of the Initiative for the Evaluation of XML retrieval (INEX). Volume 3493 of Lecture Notes in Computer Science., Schloss Dagstuhl, Germany, Springer-Verlag (2005)
[9] Trotman, A., Sigurbjörnsson, B.: Narrowed Extended XPath I (NEXI). [8]
[10] Tannier, X., Girardot, J.J., Mathieu, M.: Analysing Natural Language Queries at INEX 2004. [8] 395–409
[11] Geva, S.: GPX - Gardens Point XML Information Retrieval at INEX 2004. [8]
[12] Strzalkowski, T., ed.: Natural Language Information Retrieval. Kluwer Academic Publisher, Dordrecht, NL (1999)