Natural Language Queries for Information Retrieval in Structured Documents

Xavier Tannier, Jean-Jacques Girardot and Mihaela Mathieu
École Nationale Supérieure des Mines
de Saint-Etienne
158 Cours Fauriel
F-42023 Saint-Étienne cedex 2
Email: tannier@emse.fr, girardot@emse.fr, mathieu@emse.fr

***Abstract*— Information Retrieval in structured documents (and particularly XML) requires the user to have a good knowledge of the document structure and of some query language. This article discusses the advantages that could be brought by a system allowing natural language queries, and presents a technique to translate such requests into a formal query language.**

I. Introduction

Information Retrieval applied to documents available through the Web has provided virtually every human being the ability to find information from the very large information set constituted by search engines like Google or AllTheWeb. These engines index different document formats (text, html, etc.) which share the characteristic of being *flat*[1]: they constitute pure data, and do not embed any structure semantic information. An information request consists of a few words, and a document is considered relevant if it contains these words.

However, a new standard has now emerged, eXtensible Markup Language (XML [1]), which is more and more widely used to store and exchange information. XML files offer the particularity to provide both *information* and *structure*, since information is embedded into tags that structure it. XML therefore offers the possibility to give a rich structure to a document so that each word may have different semantics depending on the structure element which contains it.

This article discusses the benefits that can be gained from using some natural language processing techniques and the specificities of structured documents, in order to retrieve some information from an XML corpus. It also presents a technique of natural language query analysis that relies on the structure of the documents to "understand" the semantics of the request, and discusses further thoughts that need to be given in order to improve this kind of work.

II. XML and XML retrieval

XML is a text-based markup language where data is identified by nested elements enclosed in *start tags* and *end tags* that give information about the *semantics* or the *form* of the text (figure 1).

[1]Even when the document is structured (html), these systems index the textual content and do not offer the possibility to query about the structure of the document.

```
<article title="">
  <title>
    Natural Language Queries for IR...
  </title>
  <introduction>
    <bold>Information Retrieval</bold>
applied to documents available through the
<italics>Web</italics> ...
  </introduction>
  <chapter n="1">
    ...
  </chapter>
...
</article>
```

Fig. 1. example XML document

XML documents may be split into two categories:

- The *document-centric* documents where the markup gives the logical structure but where text is primordial (*e.g.*: books or articles)
- The *data-centric* documents, mainly used for machine consumption, where markup and text are indissociable (*e.g.:* flight schedules, on-line catalogues)

Information Retrieval (IR) in XML documents is quite interesting because we can separate structure and content semantically; if a user knows some information about the structure semantics, she can use it and so mix content and structure in her query. She can also ask for specific parts of the document, something that usually cannot be done for flat documents. For now, current search engines can index XML documents, but do not take into account the structures of these documents; therefore, the unique possibilities for powerful queries on structures are lost.

Although XML is recent, its usage is steadily growing, and many query languages, such as Lorel [2], XQL [3], XML-QL, XML-GL [4], Quilt [5], XQuery [6], and even XSLT [7], have already been devised for XML documents. However, the semantics of these query languages are quite complex, and their use is much more complicated than writing requests

for flat documents, or even SQL requests. The use of such languages is limited to persons who have a good knowledge of their features.

In opposition to relational databases designed for a professional use with predictable and repetitive queries, XML collections are used and will be used more and more often by large groups of users who have unpredictable and complex needs of information, and who do not know how to use sophisticated query languages.

III. How can natural language processing help?

The field of Natural Language Processing (NLP) has been extensively studied in the context of Information Retrieval over textual (flat) collections (for overviews on this subject, see [8]–[13]). Linguistic analysis of the corpus and/or the query seem to be able to carry out decisive improvements in the retrieval process, but the actual results are not yet up to those expectations [13], [14].

However we think that the spread of XML corpora can bring new hopes for at least two reasons:

- The benefit we can gain from using natural language is much higher in XML retrieval than in traditional IR. While in the last case a flat keyword-based query is quite easy to write, in XML retrieval we are confronted with two major difficulties:
  - we cannot expect casual users to learn complex and fast changing query languages like XQuery;
  - these languages also require the user to have a full knowledge of the corpus structure and of the tag names (the DTD[2]).

  Although these languages are necessary at some steps of the retrieval process (in order to actually extract the answers), we need more simple interfaces.
- To perform a really effective natural language-based retrieval in a flat text, we should "understand" the semantic of this text, which is not feasible yet. In the case of structured documents, we consider that a well-thought and semantically strong structure, because it formally marks up the meaning of the text, can make easier the query "understanding", at least when this query refers (partly) to the structure.

IV. Description of our approach

Our aim is two-fold:

- First, from a natural language query, to generate a query in a formal structured (XQuery-like) query language[3].
- Then, from this formal query, to perform a "usual" retrieval process to get answers from the corpus.

This paper focuses on the first stage, which constitutes the most innovative part of our work.

To achieve this goal, the steps to be performed are:

- a part-of-speech tagging of the query (IV-A);
- a syntactic/semantic analysis of the query (IV-B);
- with the help of specific rules (IV-C):
  - 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 *author* if it exists);
  - and a distinction between the semantical elements mapping on the structure and, respectively, mapping on the content;
- a treatment of relations existing between different elements recognized at last stage (IV-D);
- the construction of a formal language query (IV-E).

A. Part-of-speech tagging

A part-of-speech (POS), or word class, is the role played by a word in the sentence (*e.g.:* noun, verb, adjective...). POS tagging is the process of marking up the words in a text with their corresponding roles. To carry out this task we chose the free tool TreeTagger [15]. For example, for the following query:

(1) *Find the title of articles that deal with semantics*

...the output of TreeTagger is given in figure 2[4].

Find	VERB(IMP)	find
the	DET	the
title	NOUN	title
of	PREP	of
articles	NOUN(PLUR)	article
that	REL_PRO	that
deal	VERB	deal
with	PREP	with
semantics	NOUN	semantics

Fig. 2. POS tagging of sentence (1) with TreeTagger[4]. *Find* is an imperative verb, *of* and *with* are prepositions and *that* is a relative pronoun

B. Syntactic/semantic analysis

This analysis is performed with a set of rules describing the grammatical constructions that are the most current in queries and questions. These rules are said *context-free*: they define which sequence of elements (on the right side) is needed to compose a single new element (on the left side). The whole set of rules is a *context-free grammar (CFG)*. As an example, we listed in figure 3 the rules that will be triggered when parsing our sample query (1).

A recursive application of these CFG rules results in the *syntactic tree* represented in figure 4[5].

[2]The Document Type Definition (DTD) is a document defining the tags that can be used in the XML file as well as their structure (imbrication, number, sequences, etc.). A similar purpose is also achieved by the newer XSchemas, which use a more complex syntax than DTDs and are currently less used.

[3]In the whole paper we will call a *formal query language* a language with formalized semantics and grammar, as opposed to natural language.

[4]For a clearer comprehension by a non-expert reader, we changed the names of the tags into less complete but more explicit abbreviations.

[5]Note that with this set of rules two different parsing are possible: the relative proposition can be attached to the noun *"article"* (as shown in the figure) or to the noun *"title"*. In practice both trees are explored.

```
NP → DET? NOUN
NP → NP PREP NP
NP → NP REL_PROP
REL_PROP → REL_PRO VP
VP → VERB PREP? NP
S → VERB(IMP) NP
```

Fig. 3. Examples of CFG rules, with S = Sentence, NP = Noun Phrase, REL_PROP = relative proposition, VP = Verbal Phrase. The question mark "?" means that the element is optional in the sequence.

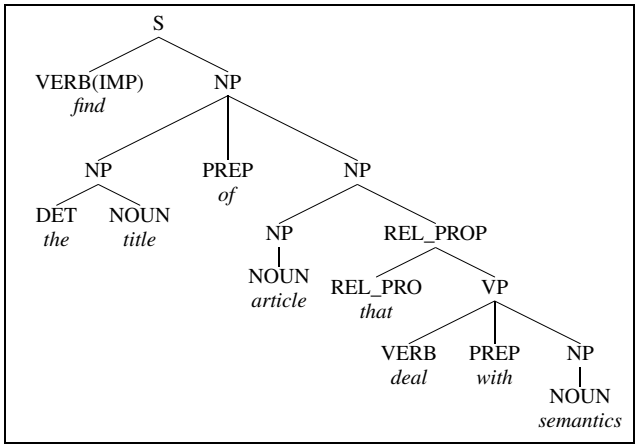

Fig. 4. Syntactic tree of sentence (1), obtained with rules of figure 3

This operation gives us a syntactic structure, but we need some semantics to have an idea about the relations existing between the words. In that aim, we use a very simple implementation of Discourse Representation Theory (DRT) [16]. In DRT, the semantic representation of a discourse (or a part of discourse) is described with a two-level "box" called "Discourse Representation Structure" (DRS). The upper level gives the discourse referents, which are the elements introduced by the discourse; the lower level represents the conditions concerning the referents.

Figure 5 shows a typical example of DRS.

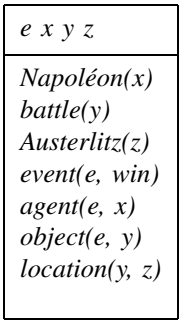

Fig. 5. Semantic representation in DRT of the sentence: *"Napoleon wins a battle in Austerlitz"*

In this example, the terms *"Napoléon"*, *"Austerlitz"*, *"battle"* and the verb *"to win"* (event e) are discourse referents, represented by letters in the upper level and described by logical predicates in the lower level. The other conditions are about the agent and object of the event (respectively *"Napoléon"* and *"battle"* for the event *"to win"*) and the location of the battle.

One can note that the semantics of some predicates can differ from a domain to another. If the *location* is here understood as a geographic feature, in an XML context it would probably be a structural constraint (in the document). Again a task-specific choice has to be made in order to model constructions that are peculiar to XML retrieval.

To compute a DRS representing a whole sentence, we attribute a basic DRS to each word, depending on its class (POS). Let us come back to our running example of sentence (1). Figure 6 contains three examples of POS DRSs for a noun, a verb and a preposition.

The syntactic rules are enriched with semantic actions. In our example, with the rule

```
VP → VERB  PREP?  NP,
```

applied to the DRSs of figure 6, we add the following identity conditions: $e_1 = e_2$ and $x = y$. The result is the semantic tree of figure 7.

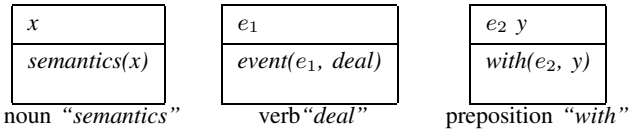

Fig. 6. examples of word DRSs

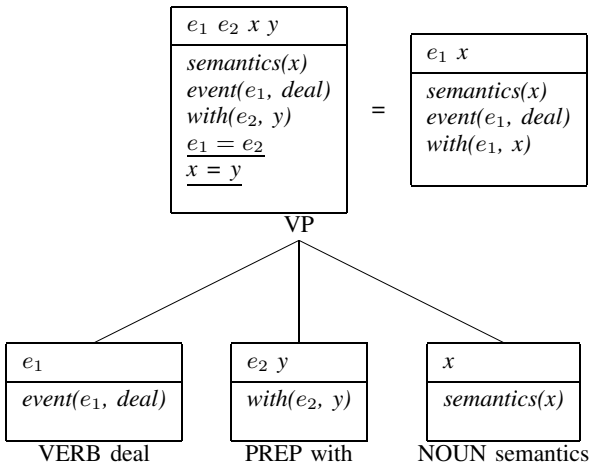

Fig. 7. example of DRS for the verbal phrase "*deal with semantics*"

Due to a lack of place we cannot show the semantic tree obtained for the whole example. The figure 8 gives the final DRS. Note that a referent has been added, which is implicit in the sentence: the *interlocutor* to which we give an order when using a verb in the imperative mood (here, *"find..."*).

$x\ y\ z\ s\ e_1\ e_2$
event(e_1, *find*)
event(e_2, *deal*)
title(*x*)
article(*y*)
semantics(*z*)
interlocutor(*s*)
agent(e_1, *s*)
object(e_1, *x*)
of(*x*, *y*)
agent(e_2, *y*)
with(e_2, *z*)

Fig. 8. DRS for sentence (1)

N.B.: The set of syntactic/semantic rules that we use is made up of about 50 rules and is obviously not intended to describe the whole language. The stress has been put on noun phrases, that are often much more meaningful than verbal phrases (at least in terms of Information Retrieval). Relative propositions, prepositional phrases are also very important because they mark a structure of query that we do not want to miss. For complex demands, an entire parsing is often impossible. In that case only the noun phrases are analyzed and the verbs are left out.

The DRS that is obtained at this stage cannot be used to build a formal query yet. Some IR-specific rules have to be set up.

C. Specific rules

The semantic construction can be reduced by taking some special cases into account, among which:

1) The **"query verbs"** like *"to want"*, *"to find"*... With the help of a dictionary describing the semantic relation between those verbs and the queries, we set a particular flag on the concerned element. This flag means that this element should be selected as a good response to the query, as shown in the following example.

   (2) I **want** an article.

s x e		
speaker(*s*) *article*(*x*) *event*(*e*, *want*) *agent*(*e*, *s*) *object*(*e*, *x*)	⇒	[*x*] *article*(*x*)

   Here we know that the verb *"to want"* means that its object (*"article"*) has to be selected, and this new information is represented by a framed referent. The agents of such verbs (here the *speaker*, or *"I"*), as well as the verbs themselves, are then left out.

2) The **description verbs** like *"to deal with"*, *"to concern"*... An other dictionary allows to add a new relation called *about*:

   (3) an article that **deals with** semantics.

x y e		
article(*x*) *semantics*(*y*) *event*(*e*, *deal*) *with*(*e*, *y*)	⇒	*x y* *article*(*x*) *semantics*(*y*) *about*(*x*, *y*)

   The verb referent is removed as well in this case.

3) The **verbs of topological relation** like *"to contain"*, *"to include"*... If such a verb has an agent and an object, then an appropriate relation is set up between those two elements and the verb is deleted:

   (4) a play that **contains** a scene...

x y e		
play(*x*) *scene*(*y*) *event*(*e*, *contain*) *agent*(*e*, *x*) *object*(*e*, *y*) ...	⇒	*x y* *play*(*x*) *scene*(*y*) *contains*(*x*, *y*) ...

4) The **words or phrases in quotation marks**[6] are considered as non-separable expressions and are grouped together in a single variable.

   (5) I am looking for the *"To be or not to be"* scene in *Hamlet*.

5) And above all, **a term recognized as a DTD-tag (or synonym)** is changed into this tag name and is marked as such (here by a bold predicate in the DRS, with *ttl* standing for *"title"* and *art* for *"article"*).

   (6) the **title** of an **article**...

x y		
title(*x*) *article*(*y*) *of*(*x*, *y*)	⇒	***x y*** ***ttl*(*x*)** ***art*(*y*)** *of*(*x*, *y*)

   A dictionary of synonyms is used. This dictionary is absolutely not intended to be general, but corpus-specific; indeed the DTD tag names are rarely real words, but abbreviations instead (*art* for *article*, *st* for *section title*, *p* or *par* for *paragraph*, etc.).

Figure 9 shows the application of some of these specific rules on our sample DRS. Let us remind the initial request:

(1) *Find the title of articles that deal with semantics*

[6]These phrases in quotation marks are detected after the morpho-syntactic analysis by a specific post-process.

We suppose that our dictionary tells us that the words *"title"* and *"article"* respectively stand for the tag names *ttl* and *art*.

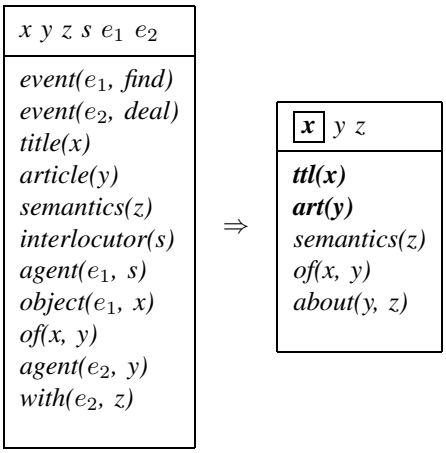

Fig. 9. DRS for sentence (1) before and after application of specific rules. Let us remind that we are looking for "a title of article that deals with semantics". The referent x is selected (rule 1), the article y is *about "semantics"* (rule 2) and the terms *"article"* and *"title"* is recognized as tag identifiers *ttl* and *art* (rule 5).

In the new DRS, we can clearly distinguish a *tag name*, which is related to the structure of the document (in bold type), from what we will now call a *term*, as *"semantics"*, which is supposed to be a part of the textual contents of the document.

D. Structure analysis

At this stage we still have some binary relations between referents that have not been treated by any specific rule (in figure 9 the relation *of(x, y)*). These relations have all a particular meaning, and a system cannot have the knowledge of each of these meanings. We implemented a semantic representation of some important relations (and particularly "temporal" relations – *after*, *included*, etc., that should be understood here as order constraints in the XML file).

Let $R(x, y)$ be a binary relation between referents x and y. To handle "known" relations as well as "unknown" ones, we apply a heuristic according to the following cases:

1) x and y refer to two tag names (representing structural elements):
   a) if the relation R is known, no action is needed.

   e.g.: A **paragraph** after a **figure**:

x y
par(x)
fig(y)
after(x, y)

   b) if the relation R is unknown, the fact that a relation exists is in itself an information: the structure given by the DTD allows then to guess which relation(s) it can be. In our example the DTD will tell us that an element *ttl* (title x) is *contained by* an element *art* (article y).

e.g.: A **title** of **article**:

x y
ttl(x)
art(y)
of(x, y)

$\xrightarrow[analysis]{DTD}$

x y
ttl(x)
art(y)
contains(y, x)

2) The relation links a tag (let us say x) and a term (y):
   a) if R is known, we add a tag with any name (called '*') that contains y, and the relation is transfered to this new tag:

   e.g.: A **paragraph** before the *conclusion* [7]:

x y
par(x)
conclusion(y)
before(x, y)

   →

x y z
par(x)
conclusion(y)
****(z)***
contains(z, y)
before(x, z)

   The paragraph should be written in the XML file before a tag that contains the word *"conclusion"*.
   b) if R is unknown, our first experiments show that a treatment produces more noise than useful information. The relation is suppressed[8].

   e.g.: A *short* **article**:

x y
art(x)
short(y)
rel_adjective(x, y)

   $\xrightarrow{remove}$

x y
art(x)
short(y)

   N.B.: This example shows that a deep semantic knowledge could be helpful. A description of what is considered to be a *short* article would allow to get more appropriate answers. But this kind of knowledge is very hard to model because of the number and the subjectivity of the relations involved.
3) R holds between two terms (*term* is here used as opposed to *DTD tag names*): in that case the relation does not apply to the structure, as x and y refer to content elements. We do not have any particular treatment to do, but the relation can be a useful linguistic information that we keep for the retrieval process (which can use it or not).

   e.g.: A *cruise* on the *Danube*:

x y
cruise(x)
Danube(y)
on(x, y)

[7] We suppose in this example that the word *"conclusion"* cannot be assimilated to a tag name.

[8] Here the referent y does not have any more relation with other elements and becomes useless, but it could be different.

If the search engine is able to handle such relations, it is useful to know that the whole phrase *"cruise on the Danube"* is preferable to the separate words *"cruise"* and *"Danube"*.

In our running example only the rule 1b applies and our final DRS is shown in figure 10.

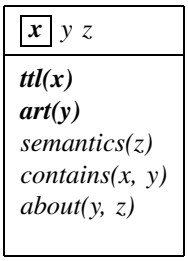

Fig. 10. final DRS for sentence (1).

E. Formal language query

At the end of the linguistic phase, our aim is to obtain a formal language query that could easily be translated into an existing XQuery-like language. From those languages (and initially from SQL) we take the idea of clause pattern (SELECT-FROM-WHERE in SQL) for restructuring the request. We chose the following four-clause pattern:

- the `from` clause contains the tag names and indications about the tag path (XPath [17] expression);
- in the `select` clause we find the elements that are to be returned to the user; those elements must be refered in the `from` clause.
- the `where` clause contains the relations between tags or variables (*e.g.:* before(x, y), about(a, b))
- the `variables` are identifiers replacing terms in the other clauses;

The transformation process from DRS to formal language is straightforward: the tag names are already flagged (in bold type in our representation), as well as the selected elements (framed referents in the DRS). The variables are the unary predicates that are not tag names, and the `where` clause corresponds to the other conditions.

The figure 11 shows a SQL-like formal representation that we can obtain from the DRS of figure 10.

```
FROM /art y,
     y/ttl x,
WHERE about(y, z)
VARIABLES z = "semantics"
SELECT x
```

Fig. 11. Example of formal query obtained from DRS 10 (query (1)). `y` is an *art* tag (article), `x` is a *ttl* tag (title) contained in `y`, and `z` is a variable representing the term *"semantics"*.

V. Comments and further work

A. About the retrieval process

At this stage we have a formal query that can be translated into an existing and implemented language such as XQuery. But while doing that we should keep in mind that when we use XQuery[9], we are not doing Information Retrieval, but rather some kind of database exploration. The differences are the following:

- In our example the *about* relation obtained from query (1) has no defined semantics or certainly no equivalent in XQuery; in order to translate it we could only require the word *"semantics"* to be present in the text, which is obviously not enough.
- As the request is a natural language query, the `from` clause should not necessarily be considered as strict paths. If the user says *"give me a paragraph about semantics"*, a section or a figure can also be relevant (maybe not even *less* relevant).
- XQuery gives us a boolean result without indication about its level of relevance. In IR we like to be able to rank the retrieved documents in relevance order.

For these reasons we think that a real IR-oriented search engine, rather than an XQuery processor, should be "plugged" there.

B. Corpus knowledge

During the stages that we detailed in the previous sections, we used several kinds of information about the explored corpus. This knowledge has to be modeled for each new set of documents[10] to study, and therefore prevents the spread of such a technique to arbitrary kinds of XML documents; this is why we tried to reduce this necessity to the minimum. But despite this, we think that the following points are the minimum knowledge to know in order to perform an appropriate query analysis:

- The Document Type Definition (DTD) that gives the common structure of the documents. To have this definition is the *sine qua non* condition; indeed a good part of the process relies on the description of the documents structure.
- In the (frequent) case where the DTD tag names are not "real" words (see section IV-C), we also need a dictionary giving the DTD semantics, because we consider that the user does not have a deep knowledge of the DTD.
- Eventually a dictionary of acceptable synonyms for the tag names (*e.g.:* paper = article = document, etc.);
- Semantic locutions (*e.g.:* "a list of keywords" = "keywords"), in order to avoid a noise generated by an erroneous detection of terms (here, *list* is neither a tag name nor a term to look for in the text);

[9]When we use the term "XQuery" it should be understood as "XQuery-like query languages".

[10]We call a *new* set of documents a corpus with a different DTD and different subject.

- Some very simple ontologic structures (*e.g.:* "a novel written by Marcel Proust" = "a novel of which the author is Marcel Proust" – if we suppose that *novel* and *author* represent tag names);

Obviously, the more knowledge we have about the corpus, the best the query analysis can be. For example an ontology of the domain treated by each document could also help. But the points we enumerated describe a knowledge that we think is hard to avoid.

We think that it is necessary to give further thoughts to the problem of the antagonism between re-usability of a technique on the one hand, and corpus- or domain-dependant knowledge used on the other hand. The work around Semantic Web [18] tends to the idea of providing a common framework to define data structure and meaning, and moves towards a complete and precise semantic description of the document.

If some knowledge modelling seems inescapable, the fact is that a real development of XML retrieval depends on how easy the installation of a system in a new context is. It seems that the good balance does not only depend on the different kinds of knowledge we consider as necessary, but also on the way we model this knowledge and on the corpus category (document- or data- centric).

VI. Conclusion

Natural language querying in XML corpora should contribute to the expansion of structured document engineering, by allowing any non-expert user to search some information in these documents, which are more and more numerous in companies and on the Internet. A lot of work has to be done to achieve this goal. In this article, we have described a technique to translate a natural language request into a formal language query, with the help of information about the structure of the documents described in the DTD.

References

[1] "Extensible Markup Language (XML). World Wide Web Consortium (W3C) Recommandation," http://www.w3.org/TR/REC-xml/.

[2] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom, "Lore: A Database Management System for Semistructured Data," *SIGMOD Record*, vol. 26, no. 3, pp. 54–66, Sept. 1997.

[3] J. Robie, J. Lapp, and D. Schach, "XML Query Language (XQL)," http://www.w3.org/TandS/QL/QL98/pp/xql.html.

[4] S. Ceri, S. Comai, E. Damiani, P. Fraternali, S. Paraboschi, and L. Tanca, "XML-GL: a Graphical Language for Querying and Restructuring XML Documents," in *Proceedings of the 8th International WWW Conference, WWW8*. Toronto, Canada: International World Wide Web Conference Committee (IW3C2), May 1999.

[5] D. Chamberlin, J. Robie, and D. Florescu, "Quilt: An XML Query Language for Heterogeneous Data Sources," in *Proceedings of WebDB 2000 Conference*, ser. Lecture Notes in Computer Science. Springer-Verlag, 2000.

[6] "XQuery 1.0: An XML Query Language. World Wide Web Consortium (W3C) Working Draft," http://www.w3.org/TR/xpath.

[7] "XSL Transformation (XSL). World Wide Web Consortium (W3C) Recommandation," http://www.w3.org/TR/xslt/.

[8] A. F. Smeaton, "Information Retrieval: Still Butting Heads with Natural Language Processing?" in *Information Extraction – A Multidisciplinary Approach to an Emerging Information Technology*, ser. Lecture Notes in Computer Science, M. Pazienza, Ed. Springer-Verlag, 1997, vol. 1299, pp. 115–138.

[9] S. Feldman, "NLP Meets the Jabberwocky: Natural Language Processing in Information Retrieval," Online, May 1999, http://www.onlinemag.net/OL1999/feldman5.html.

[10] A. F. Smeaton, "Using NLP or NLP Resources for Information Retrieval Tasks," in *Natural Language Information Retrieval*, T. Strzalkowski, Ed. Dordrecht, NL: Kluwer Academic Publisher, 1999, pp. 99–111.

[11] A. Arampatzis, T. van der Weide, C. Koster, and P. van Bommel, "Linguistically-motivated Information Retrieval," in *Encyclopedia of Library and Information Science*, A. Kent, Ed. New York, Basel: Marcel Dekker, Inc., Dec. 2000, vol. 69, pp. 201–222.

[12] C. Jacquemin and P. Zweigenbaum, "Traitement automatique des langues pour l'accès au contenu des documents," in *Le document en sciences du traitement de l'information*, T. Cepadues, Ed. Jacques Le Maître, Jean Charles et Catherine Garbay, 2000, ch. 4, pp. 71–109.

[13] K. S. Jones, "What is the role of NLP in text retrieval?" in *Natural Language Information Retrieval*, T. Strzalkowski, Ed. Dordrecht, NL: Kluwer Academic Publisher, 1999, pp. 1–24.

[14] T. Strzalkowski, "Preface," in *Natural Language Information Retrieval*, T. Strzalkowski, Ed. Dordrecht, NL: Kluwer Academic Publisher, 1999, pp. xiii–xxiii.

[15] H. Schmid, "Probabilistic Part-of-Speech Tagging Using Decision Trees," in *International Conference on New Methods in Language Processing*, Sept. 1994.

[16] H. Kamp and U. Reyle, *From discourse to logic*. Kluwer Academic Publisher, 1993.

[17] "XML Path Language (XPath). World Wide Web Consortium (W3C) Recommandation," http://www.w3.org/TR/xpath.

[18] "Semantic Web. World Wide Web Consortium (W3C)." http://www.w3.org/2001/sw/.

[19] T. Strzalkowski, Ed., *Natural Language Information Retrieval*. Dordrecht, NL: Kluwer Academic Publisher, 1999.