Approaches to Translating Natural Language Queries for use in XML Information Retrieval Systems

Xavier Tannier
Ecole des Mines
Saint-Etienne, France

Alan Woodley
Queensland University of Technology
Brisbane, Australia

Shlomo Geva
Queensland University of Technology
Brisbane, Australia

Marcus Hassler
Universität Klagenfurt
Klagenfurt, Austria

Abstract

XML information retrieval (XML-IR) systems aim to provide users with relevant results that are more specific than documents. To interact with XML-IR systems, users must express both their content and structural requirements in the form of a structured query, using formal languages such as XPath or NEXI. Natural language queries (NLQs) are a more intuitive alternative. Here, we present three approaches that analyse NLQs and translate them into a formal language (NEXI) query. The approaches participated in INEX's 2005 NLP track, where they performed strongly, even outperforming a baseline that consisted of manually constructed NEXI expressions. This suggests that further collaboration between NLP and XML-IR could be mutually beneficial.

1 Introduction

Information retrieval (IR) systems respond to user queries (historically two or three keywords, expressing desired content) with a ranked list of relevant documents. However, this does not guarantee user satisfaction since in many cases only some portions of documents contain relevant information. Yet, this transaction has remained unchanged since the earliest days of IR, partly since the documents retrieved by most IR systems usually contain little or

```
<article>
     <author>Roger Fuller</author>
     <title>Toward a robust Martian-English
     translator</title>
     <section title="Introduction">
          <paragraph>Because of a dramatic
          lack of interpreters, Communication
          between <bold>Martians</bold>
          and <bold>Terrestrians</bold> is
          confronted to... </paragraph>
     </section>
     <section title="Morphology">
          ...
     </section>
     ...
</article>
```

Figure 1: XML representation of a science article.

no underlying structural information (and while HTML documents contain structure, most HTML tags relate solely to document presentation rather than to semantic inference on part of the author).

In comparison with flat text documents, XML articles explicitly separate content, structure and form (see Figure 1). The recent prominence of XML markup has lead to IR systems specifically geared towards collections of XML documents. With their hierarchical structure, XML documents provide an opportunity to improve upon the traditional paradigm of IR with a new retrieval unit: the document *component* (formally marked up by tags in XML documents). The new challenge of XML-IR is then to find the best compromise between the component's degree of relevance (or *exhaustiveness*) and its *specificity* to a query topic.

Owing to these characteristics, XML-IR combines features from traditional information retrieval (IR) and from database retrieval. Like traditional IR, it responds to queries with a ranked list of relevant results, and like database retrieval (but unlike traditional IR) it requires a mean to express both content and structural needs if users are to take full advantage of XML-IR systems. For this reason, formal structured query languages (akin to SQL for databases) have been designed. Some examples of these languages are XQuery [16], XPath [15] and NEXI [14]. Unfortunately, these formal languages have proven problematic for several reasons: they are very complex and difficult to use even by experienced, let alone casual users; are too closely bound to the underlying physical structure of the collection; and

do not scale well across multiple or heterogenous collections.

Recent research has investigated the idea of using the specifics of XML retrieval to allow users to address content and structural needs intuitively via natural language queries (NLQs). Here, we present the motivations for this research (Section 2), as well as three systems that analyse NLQs and translate them to an existing formal language (NEXI – Section 3). The systems performed strongly, at times outperforming a baseline that consisted of a set of manually constructed NEXI expressions (Section 4). These results provide the best indication yet of the potential for NLQs to become a viable alternative to XML-IR systems, and, as we discuss towards the end of the paper (Section 5), indicates that further collaboration between NLP and XML-IR communities could be mutually beneficial.

2 Motivation

We are not going to detail the general motivations for applying natural language processing techniques to information retrieval. These applications have been extensively studied [11, 2, 12]. The issues outlined in this section are specific to structured IR. Some of them are also important in the domain of databases, but we will see that natural language interfaces for database and for XML collections correspond to different needs and have different traits.

The first motivation for designing natural language interfaces for XML retrieval is that expressing an information need in a structured language, with formalized semantics and grammar, is too difficult for many users. *O'Keefe and Trotman* [9] investigated five structured query languages (HyTime, DSSSL, CSS, XPath, XIRQL) and concluded that all of them were too complicated to use. During the INEX 2003 campaign, where queries were built by experts in IR, XPath [15] was used to represent the information needs, and 63% of the proposed queries had major semantic or syntactic errors, requiring up to 12 rounds of corrections. This led to the adoption of NEXI, a simplified language, in 2004. That year, the number of revisions decreased and the error rate dropped to 12%. But this is still high for experts. Whether for casual or expert users, everyday ("natural") language seems to be the easiest and most intuitive way to express an information need.

Secondly, formal query languages require an intimate knowledge of a document's structure and semantics. For instance, in order to retrieve information from abstracts, sections or bibliographic items, users need to know

whether these elements are properly identified in the structure, and what the corresponding markup tags are (for example <*abs*>, <*sec*> and <*bb*> respectively). This information is contained in the DTD or Schema, but this is another language to learn and more information to remember (for instance the INEX DTD contains 192 different content models).

Thirdly, single formal queries do not scale well for information retrieval in heterogenous collections (collections with different DTDs or Schemas) since different collections often use different markup tags for the same retrieval unit (for example: paragraph could be <*p*>, <*para*> or <*paragraph*>). A natural language interface could resolve this problem, since users could express their information need conceptually. This is then translated into several formal queries (one per DTD).

Finally, in structured documents, a well-thought and semantically strong structure formally marks up the meaning of the text; this can make it easier to "understand" queries. For example, given the keyword *'Washington'* a well structured document can more easily identify the difference between the American President, City, State or Monument. Moreover, a system can introduce appropriate structural constraints even if these constraints have not been specified by the user.

2.1 Positioning in NLP for Information Seeking

Information retrieval can accomplish the following types of missions involving a search engine [8]: *Known item search*, where users know what elements (records) they are looking for, and can recognize them if seen, is a mission carried out by *database* querying. *Specific information search*, where users want some specific information but do not necessarily know where to find it, is fulfilled by *question-answering* systems. Finally, in *general information search*, the user is interested in a subject in general, with several ways to describe it, and several ways to represent the desired information. This is the field of information retrieval. This section briefly outlines the role of NLP in these fields.

2.1.1 NLP and Information Retrieval

At first, it seems only logical that NLP and traditional IR would complement each other. However, incorporating NLP techniques within IR has been surprisingly ineffective [12]. NLP techniques are rarely used within traditional IR (with the exception of low-level techniques such as phrase indexing, stemming and spelling correction).

One possible explanation for this is that IR systems do not have to understand the queries that are submitted to them. Rather, they have to match queries to relevant documents, usually by just matching the query terms to document terms (or some variation thereof). Since most relevant documents tend to contain a large number of occurrences of query terms, statistical rather than linguistic methods have proven successful. However, information seeking areas that deal with much smaller text regions such as Question and Answering (QA) make stronger user of NLP. And since XML-IR is conceptually between traditional IR and QA, we believe that incorporating NLP techniques should also be successful.

2.1.2 NLP and Databases

Many natural language interfaces for databases have been developed, most of them transforming natural language into Structured Query Language (SQL) [1, 3, 10]. But the problems are different:

- 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 not information retrieval. The user knows what kind of information is stored in the database, the information need is precise, and a correct query necessarily leads to a correct answer. An erroneous interpretation of the request leads to totally useless results - or to no results at all. This means that the natural language analysis must interpret the query perfectly and unambiguously, failing which the final answer is incorrect and the user dissatisfied. For this reason, natural language interfaces for databases notably only apply to restricted domains (as geographic databases) with a restricted language (the answer to a query is often "I did not understand your query").

  In contrast, 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. We can even imagine an interface getting *better* results than manual queries (which makes no sense in databases).

- Moreover, information retrieval does not require any operation on data (calculation, concatenation, aggregation, restructuring...). The answer is a set of XML elements that are part of the collection.

In return, NLIs for XML-IR are expected to analyse all queries, even partially or imperfectly, written in an unrestricted natural language, and for more general applications.

Therefore, developing NLIs for XML-IR is a separate research domain requiring its own innovative solutions.

2.1.3 NLP and QA Systems

In question-answering (QA) systems users ask closed questions, requiring a short and factual answer "When did Napoleon die?" → "in 1821". Systems must select a relevant segment of text in the collection. In XML-IR the retrieval unit, although more flexible than traditional IR, is still formally delimited by the document's underlying structure, and the information need is more general, represented by open requests. Nevertheless, it is conceivable that there may be some common queries that concern small elements (such as publication date). Moreover, XML-IR could serve as an effective passage retrieval pre-processing step in QA.

2.2 INEX's NLP Track and NLQ2NEXI Task

2.2.1 INEX and NEXI

The INitiative for the Evaluation of XML Retrieval (INEX)[1] provides a framework for evaluating XML-IR systems: a test collection consisting of over 700 Mbytes of XML documents, topics and human relevance assessments.

Figure 2 is an example of an INEX topic. The format of INEX topics is based on TREC topics and contains an initial topic statement, (cas)title, description and narrative, all of which express users' information need. Currently the <*castitle*> and <*description*> elements are used, respectively, as queries in the Ad-hoc and NLP tracks. The <*InitialTopicStatement*> and <*narrative*> are used by INEX human judges during the assessment phase. The <*castitle*> represents users' information need as a formal XPath-like language (CAS) called Narrowed Extended XPath I (NEXI [14]). The <*description*> expresses users' need in a natural language (e.g. English),

[1] `http://inex.is.informatik.uni-duisburg.de/2005/`

```
<inex_topic topic_id="256" query_type="CAS">
     <InitialTopicStatement>Find Information regarding data embedding using
     watermarking.</InitialTopicStatement>
     <title></title>
     <castitle>
          //article[about(.//p,"data embedding")]
          //p[about(.,watermarking)]
     </castitle>
     <description>
          We are looking for paragraphs describing watermarking in articles which describe
          data embedding.
     </description>
     <narrative>
          In today's world the issue of data security is highly significant. One such technique
          to ensure data security is steganography where data is embedded in various media
          files [...]
     </narrative>
</inex_topic>
```

Figure 2: Example of INEX query.

and is (hopefully) a faithful translation of the title. The syntax of NEXI is similar to XPath [15], however, it only uses descendant axis step, and extends XPath by incorporating an *"about"* clause to provide IR flavour to queries. NEXI's syntax is:

$$//A[about(//B,C)]$$

where A is the context path, B is the relative path and C is the content requirement.

It is possible for a single NEXI query to contain more than one information request. Therefore the query presented in Figure 2:

//article[about(.//p, "data embedding")]//p[about(.,watermarking)]

contains two information requests (also called sub-topics):

$$//article//p[about(.,watermarking)]$$

And:

$$//article[about(.//p, "data\ embedding")]$$

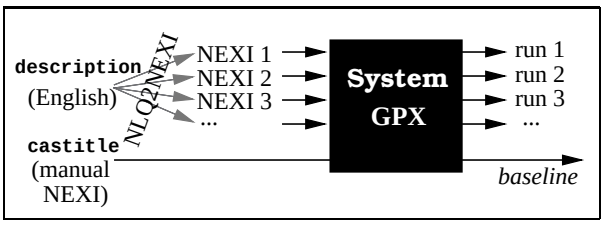

Figure 3: NLQ2NEXI.

In NEXI each information request is specified by an *'about'* clause. However, elements matching the rightmost *'about'* clause, here the first request, are returned to the user. INEX refers to these requests and elements as "target requests" and "target elements". Elements that match the other *"about"* clauses, here the second request, are used to support the return elements in ranking. We refer to these requests and elements as "support requests" and "support elements". In order to be valid, each NEXI query must have at least one target request, along with any number of support requests.

2.2.2 INEX NLP Track

The INEX NLP track has run for 2004 and 2005. In 2004, participants produced complete NLQ XML-IR systems that accepted natural language queries as input and produced a ranked list of XML elements as output. This procedure was identical to that of INEXs main Ad-hoc track, with the exception that natural language (description) rather than formal queries (castitle) were used as input. In 2005, an additional NLQ2NEXI task was added. Participants in the NLQ2NEXI task accepted natural language queries as input and produced NEXI queries as output. The NEXI queries were executed on an existing Ad-hoc XML-IR system (GPX [6]) that produced a ranked list of XML elements. Then, the result lists were evaluated using a standard module as if they were traditional Ad-hoc submissions (see Figure 3). This approach was adopted for three reasons.

First, it provided for a more meaningful comparison between systems since evaluation was made solely on how well the systems translated NLQs to NEXI. Previously, complete NLQ XML-IR systems were compared. Performance was dependent upon how well systems translated the natural language queries to a formal language, and then by how well the backend XML-IR system was able to execute the queries. This is problematic since the performance in one of the steps could obfuscate the performance in the other. Keeping the backend constant removed this problem.

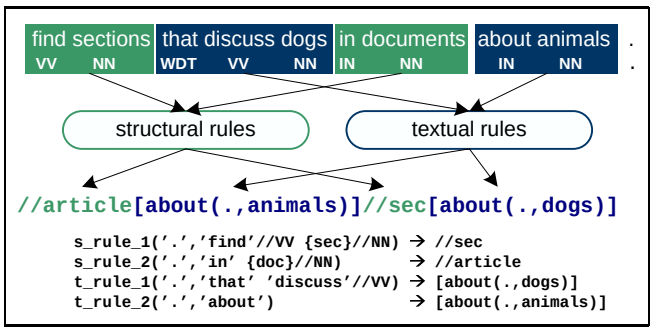

Figure 4: Query analysis with template matching.

Second, it introduced a standard baseline across all participants (here, a corresponding set of manually constructed ad-hoc track NEXI expressions). Previously, participants could only study the relative performance of NLQ systems. However, it is vital that participants are able to compare performance with a set of manually constructed NEXI expressions as it indicates the costs and benefits of using NLQs rather than formal language queries.

Third, it provided a lower cost of entry for participation. Despite the success of workshops such as TREC and INEX, some of their tracks have a relatively low number of participants. A possible explanation for this situation is that such workshops often require a considerable commitment, particularly for first time participants. Previously, participants had to produce two systems: one to handle to translation from NLQ to NEXI, and a second to accept the NEXI queries and produce results lists. But using a single backend system meant participants could focus on producing NLQ to NEXI translators.

3 The Approaches

Here, we present a selection of techniques used to translate NLQs to NEXI in INEX 2004 and 2005. While each of the approaches is different, they all contain four main stages: Detecting structural and content constraints, determining structural requirements, determining content requirements and finally NEXI query production.

3.1 Detecting Structural and Content Constraints

The first stage is to distinguish a query's structural constraints from content constraints (that will be expressed in the *'about'* clause in NEXI). *Hassler* sets up a template matching based on words and parts-of-speech (see Figure 4). Links between structural elements and content are not linguistically motivated, it is assumed that content is included in the last introduced element. This technique is very efficient for queries expressed in a traditional way (*i.e.* constructions that have been recorded by the system). It avoids syntactic ambiguities but lacks robustness. *Woodley and Geva* [17] add a shallow syntactic parsing before applying the same kind of template matching. They deal with negations, strengtheners and a few forms of anaphora. *Tannier* [13] uses a deep syntactic analysis, complemented by some specific semantic rules concerning query structure ("find + object", "deal with", etc.). This method allows treatment of (even imperfectly) non-anticipated constructions, but needs a heavy disambiguation process.

3.2 Structure Analysis

The second stage is to map structural constraints to corresponding XML markup tags. For this stage lexical knowledge about collection structure is necessary, especially since the tags in the XML documents are rarely "real" words or phrases, but rather abbreviations (<*sec*> for sections), acronyms (<*st*> for section titles) or a loose amalgamation of two (<*atl*> for article titles). Furthermore, a single tag can be referred to by different names (for example: "document", "article", "work" for a scientific article).

Grammatical knowledge can be added [13] in order to recognized some frequent linguistic constructions that implicitly refer to structure (the agent of the verb "to write" is an author (<*au*>), the object of "to cite" is a bibliographic item (<*bb*>), etc.).

Unlike database querying, XML NLIs do not require domain-dependant knowledge (although this is a potentially fruitful path to follow).

3.3 Content Analysis

The third stage is to derive users' content requirements, as either terms or phrases. Noun phrases are particularly useful in information retrieval. They are identified as specific sequences of parts-of-speech [17]. *Tannier* [**?**] is also able to use content terms to set up a *contextual search* along the entire structure of the documents (see Figure 5).

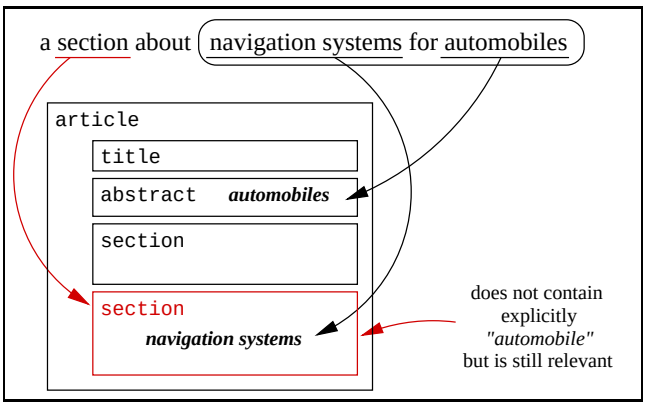

Figure 5: Contextual search.

3.4 NEXI Query Formulation

The final stage of translation is the formulation of NEXI queries. The target request is formulated first and then each of the support requests is "inserted" by finding their longest shared descendant. Following NEXI format, content terms are delimitated by spaces, with phrases surrounded by quotation marks.

4 Results

Here, we present the results from the 2005 INEX NLQ2NEXI Track. Two types of structured NLQs were used for INEX 2005: COS queries that represented a simple information need (such as *"curricula vitae about information retrieval students"*) and CAS queries that represented a more complex information need involving one or more support requests (such as the examples of queries used throughout this paper).

Evaluation in XML-IR is more complex than traditional information retrieval because relevancy between ancestors and descendants is inherently dependant, that is, if an element is relevant then so are its ancestor elements. In contrast, traditional IR assumes independence between results. Therefore, there exist several differences in assessing traditional and XML-IR.

Relevancy in XML retrieval is assessed over two dimensions: *exhaustiveness* which measures the extent to which an element satisfies the information need and *specificity* that measures the focus of the element on the information need. Relevancy is assessed on a graded (or continuous) rather than

	Baseline	*Hassler*	*Tannier*	*Woodley*
nxCG[25]				
Strict	**0.0903**	0.0807	0.0329	0.0366
Gen	**0.2541**	0.2524	0.2146	0.1617
ep-gr				
Strict	**0.0189**	0.0221	0.0124	0.0123
Gen	**0.0904**	0.0864	0.0741	0.0611

Table 1: COS, Thorough subtask

binary scale by taking the product of the two exhaustiveness and specificity scores. Here, we present the results of both the strict metric, that only rewards highly relevant and highly specific results, and the generalized metric, that also rewards intermediate results. Metrics have to reflect the graded relevance. In 2005, INEX used two metrics, the normalized extended Cumulative Gain metric (nxCG) and the Effort-Precision Gain-Recall (ep–gr) metric [7] .

For each task we compare each of the three approaches presented in the previous section, and a fourth "baseline" system which used a manually constructed NEXI expression as input. Unfortunately, due to length constraints we are unable to present all results from INEX 2005. However, here we present a summary of COS and CAS topics. A more detailed comparison is available in the *INEX 2005 Preproceedings* [4].

4.1 COS Topics

Here, we present results from two of the COS subtasks: the Thorough subtask that dealt with retrieving all elements that matched an information request, and the Focused subtask that dealt with **exclusively** retrieving the most relevant element along a path and not any of its ancestors/descendants. Tables 1 and 2 present the nxCG value at 25 results plus the ep-gr value for each subtask. These results are promising since most of the time the approaches are comparable (¿0.8) with the baseline system. Particularly promising is Hassler's approach which many times outperformed the baseline.

4.2 CAS Topics

We present results from four of the CAS subtasks. Each of the subtasks relates to how structural constraints are evaluated with respect to the target and support elements. A *strict* interpretation means that returned elements

	Baseline	*Hassler*	*Tannier*	*Woodley*
nxCG[25]				
Strict	**0.1088**	0.1313	0.1023	0.0550
Gen	**0.1998**	0.1925	0.1853	0.1698
ep-gr				
Strict	**0.0193**	0.0268	0.0130	0.0222
Gen	**0.0860**	0.0774	0.0717	0.0693

Table 2: COS, Focussed subtask

	Baseline	*Hassler*	*Tannier*	*Woodley*
nxCG[25]				
Strict	**0.1578**	0.1378	0.1378	0.1378
Gen	**0.2885**	0.3814	0.2693	0.2859
ep-gr				
Strict	**0.0770**	0.0740	0.0775	0.0755
Gen	**0.1324**	0.1531	0.1064	0.1051

Table 3: CAS, SSCAS subtask

	Baseline	*Hassler*	*Tannier*	*Woodley*
nxCG[25]				
Strict	**0.0662**	0.0662	0.0662	0.0913
Gen	**0.1081**	0.1046	0.1004	0.1100
ep-gr				
Strict	**0.0274**	0.0267	0.0304	0.0267
Gen	**0.0272**	0.0287	0.0298	0.0311

Table 4: CAS, SVCAS subtask

markup tag must exactly match the user request. In contrast, a *vague* interpretation means that the element type does not need to exactly match what the users requested. Each of these interpretations can be applied to target and support elements in isolation, producing four subtasks called SSCAS, SVCAS, VSCAS, VVCAS where the first character refers to the target element interpretation (S)trict or (V)ague, and the second character refers to the support element. Tables 3 to 6 present the nxCG value at 25 results and the ep-gr value for each subtask. Again, the NLP approaches perform comparably to – and many times outperforming – the baseline. This demonstrates the potential for NLIs as an alternative to formal language interfaces in XML-IR.

	Baseline	*Hassler*	*Tannier*	*Woodley*
nxCG[25]				
Strict	**0.1267**	0.1133	0.1133	0.1133
Gen	**0.2531**	0.2815	0.3051	0.2446
ep-gr				
Strict	**0.0383**	0.0338	0.0363	0.0340
Gen	**0.0608**	0.0641	0.0682	0.0632

Table 5: CAS, VSCAS subtask

	Baseline	*Hassler*	*Tannier*	*Woodley*
nxCG[25]				
Strict	**0.1267**	0.1267	0.1867	0.1644
Gen	**0.2281**	0.2456	0.2572	0.2136
ep-gr				
Strict	**0.0454**	0.0372	0.0418	0.0483
Gen	**0.0694**	0.0740	0.0799	0.0742

Table 6: CAS, VVCAS subtask

5 Discussion

Previously, we have discussed the "state–of–the–art" for NLP and XML-IR. Here, we make a logical progression to discuss the future of NLP and XML-IR, and how their continual integration will be mutually beneficial.

5.1 NLP and XML retrieval

The INEX NLP Track provides an opportunity for reviving the applications of NLP in the field of IR. Historically, NLIs have been shown to be effective in handling information needs. We believe that the specificities of XML documents and their effects on IR will generate further innovations in search interfaces. Furthermore, the combination of NLP and XML-IR will also improve the retrieval process itself, as illustrated by the use of noun phrase analysis for contextual search; this approach has shown promising results. Finally, since XML-IR provides the possibility to retrieve very specific texts from a large set of documents it can be helpful for traditional NLP fields such as question-answering or automatic summarization.

5.2 NEXI extensions

Extensions of NEXI are planned in 2006 to introduce new features for query formulation. The extensions aim at encouraging researchers from other fields like NLP or ontology generation to participate in future NLP tasks of INEX. It allows appending additional information to structural and content constraints using a set of *key*:*value* pairs. Applied to structural constraints, it can be used to express the type of matching (e.g. `//section{MATCH:strict}`). Furthermore, it offers the possibility to include linguistic knowledge to query terms and phrases. Thus, part-of-speech tags (e.g. `book{POS:NN}`), or morpho-syntactic information (e.g. `look{TIME:pres}`) or semantic information as ontology/domain knowledge (e.g., `dog{SEM:anim}`) can be expressed.

6 Conclusion

While the application of NLP XML-IR is in its infancy, it has already produced promising results. Here, we presented the specificities of XML retrieval and the necessity for creating NLIs for XML-IR systems. We surveyed the different approaches that have been implemented so far, described their results in the INEX 2005 campaign and discussed future directions of this field. The results show that NLQs are potentially a viable alternative to formal query languages and the integration of NLP and XML-IR can be mutually beneficial.

References

[1] I. Androutsopoulos, G.D. Ritchie, and P. Thanisch. Natural Language Interfaces to Databases – An Introduction. *Journal of Natural Language Engineering*, 1(1):29–81, 1995.

[2] Avi Arampatzis, Th.P. van der Weide, C.H.A. Koster, and P. van Bommel. Linguistically-motivated Information Retrieval. In Allen Kent, editor, *Encyclopedia of Library and Information Science*, volume 69, pages 201–222. Marcel Dekker, Inc., New York, Basel, December 2000.

[3] Ann Copestake and Karen Sparck Jones. Natural Language Interfaces to Databases. *The Knowledge Engineering Review*, 5(4):225–249, 1990.

[4] Norbert Fuhr, Mounia Lalmas, Saadia Malik, and Gabriella Kazai, editors. *Advances in XML Information Retrieval and Evaluation: Fourth*

Workshop of the INitiative for the Evaluation of XML Retrieval (INEX 2005), volume 3493 of *Lecture Notes in Computer Science*, Schloss Dagstuhl, Germany, November 28-30, 2005, 2006. Springer-Verlag, New York City, NY, USA.

[5] Norbert Fuhr, Mounia Lalmas, Saadia Malik, and Zoltàn Szlàvik, editors. *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, December 6-8, 2004, 2005. Springer-Verlag, New York City, NY, USA.

[6] Schlomo Geva. GPX - Gardens Point XML Information Retrieval at INEX 2004. In Fuhr et al. [5], pages 211–223.

[7] Gabriella Kazai and Mounia Lalmas. INEX 2005 Evaluation Measures. In Fuhr et al. [4], pages 16–29. http://inex.is.informatik.uni-duisburg.de/2005/inex-2005-metricsv4.pdf.

[8] Charles T. Meadow, Bert R. Boyce, and Donald H. Kraft. *Text Information Retrieval Systems*. Academic Press, New York City, NY, USA, San Diego, second edition, 2000.

[9] Richard A. O'Keefe and Andrew Trotman. The Simplest Query Language That Could Possibly Work. In Norbert Fuhr, Mounia Lalmas, and Saadia Malik, editors, *Proceedings of the second Workshop of the Initiative for the Evaluation of XML retrieval (INEX), December 15–17, 2003*, pages 167–174, Schloss Dagstuhl, Germany, 2004.

[10] C.R. Perrault and B.J. Grosz. Natural Language Interfaces. *Exploring Articial Intelligence*, pages 133–172, 1988.

[11] Alan F. Smeaton. Information Retrieval: Still Butting Heads with Natural Language Processing? In M.T. Pazienza, editor, *Information Extraction – A Multidisciplinary Approach to an Emerging Information Technology*, volume 1299 of *Lecture Notes in Computer Science*, pages 115–138. Springer-Verlag, New York City, NY, USA, 1997.

[12] Karen Sparck Jones. What is the role of NLP in text retrieval? In Tomek Strzalkowski, editor, *Natural Language Information Retrieval*, pages 1–24. Kluwer Academic Publisher, Dordrecht, NL, 1999.

[13] Xavier Tannier, Jean-Jacques Girardot, and Mihaela Mathieu. Analysing Natural Language Queries at INEX 2004. In Fuhr et al. [5], pages 395–409.

[14] Andrew Trotman and Börkur Sigurbjörnsson. Narrowed Extended XPath I (NEXI). In Fuhr et al. [5], pages 16–40.

[15] XML Path Language (XPath). World Wide Web Consortium (W3C) Recommandation, 1999. http://www.w3.org/TR/1999/REC-xpath-19991116.

[16] XQuery 1.0: An XML Query Language. World Wide Web Consortium (W3C) Working Draft, 2005. http://www.w3.org/TR/2005/WD-xquery-20050404/.

[17] Alan Woodley and Schomo Geva. NLPX at INEX 2005. In Fuhr et al. [4].