

XOR – XML Oriented Retrieval



Shlomo Geva, Marcus Hassler, and
Xavier Tannier

(QUT, Universitat Klagenfurt, Ecole
Nationales Supérieure des Mines)

XML and Database Retrieval vs. IR

- ❑ *Standard* tools for Database Retrieval do not support text-based IR adequately
- ❑ *Standard* tools for XML Retrieval do not support text-based XML-IR adequately
- ❑ XML-IR (a la INEX) has the potential to bridge the historical divide between the vague text-based IR and the strict structure-based DB and XML technology.

NEXI – Narrowed Extended XPath I

- ❑ Designed to address the over-kill of XPath in relation to IR needs
- ❑ Eliminated most of the complexity associated with node selection
- ❑ Introduced the about() function (explicit vagueness)
- ❑ Prescribed loose interpretation of structural constraints (implicit vagueness)

NEXI in practice

- ❑ NEXI was successfully used at INEX 2004 and 2005
- ❑ Did it improve results over keyword based search?
 - Definitely for some topics
 - Not for all topics
 - Only improved for some systems
 - On average - not by much if at all
- ❑ Why did it not improve across the board?

Natural Language Queries

- ❑ Introduced at INEX 2004, run 2004-2006
- ❑ The objective is to translate a natural language query (the <description>) into NEXI
- ❑ Why NLQ and why NLQ2NEXI?
 - Test the Hypothesis that natural language may be quite an appropriate and effective way to express structural and content requirements/constraints
 - Low entry cost into INEX, without the need to develop a search engine, but still allowing for the critical IR-craft of query formulation and expansion

The NLQ2NEXI experience

- ❑ The results that were obtained in 2004 and 2005 are encouraging
- ❑ The performance of NLQ2NEXI systems, by comparison with the baseline, is quite reasonable (a tradeoff)
- ❑ However NEXI is not able to support the NLQ technology due to:
 - imposed restrictions
 - missing functionality.

XOR (originally NE²XI)

- ❑ Designed from the outset as an open and extensible IR oriented language
- ❑ Still very simple – does not complicate NEXI to any great extent
- ❑ The XOR query specification, in the spirit of NEXI (and IR), is just a “wish list”
 - it is left to the implementation to interpret it
 - NEXI queries are XOR compatible
 - XOR queries are easily interpreted as NEXI

XOR vs. NEXI

- Example – “looking for information relating to Einstein’s 1905 paper concerning electrodynamics”

- NEXI:

//article[.//year=1905

AND about(.//author, Einstein)

AND about(.//*, electrodynamics)]

XOR vs. NEXI

□ XOR:

./year=1905



//article[about(./year,1905)
AND about(./author, Einstein)
AND about(./*, electrodynamics)]

OR

//article[about(.,Einstein article 1905)
AND about(., electrodynamics)]

OR

//article[about(./year,1900)
AND about(., electrodynamics)]

XOR vs. NEXI

- ▣ Supporting the NOT operator.

//A[about(.,B)] AND NOT
(//A[about(.,C)] or //A[about(.,D)])

- ▣ Only as a qualifier to other “positive” selection

~~NOT //A[about(.,B)]~~

XOR qualifiers {...}

- ❑ XOR introduces a simple syntax for specifying qualifiers to query elements

```
//article[about(./year{mode:strict},1905)  
    AND about(./author, Einstein)  
    AND about(./*, electrodynamics)]
```

```
//abstract{mode:vague}[about(.,GO{POS:Noun})]
```

```
//section{mode:vague}[about(.,AJAR{CASE:upper})]
```

```
//article[about(.,Germany) AND{mode:strict}  
    about(.,football)]//sec[about(.,Europe)]
```

XOR and hyperlinks

- ❑ Two functions are proposed, **LinkTo()** and **LinkFrom()**
- ❑ Both have about() semantics -
 - **LinkTo(LinkNode, keywords)**
checks that the linked-to element is **about** the keywords.
LinkNode specified in standard format (XLink/XPointer)
 - **LinkFrom(ContextNode, keywords)**
checks that the linked-from element
is **about** the keywords
- ❑ This is clearly more difficult to support than the vanilla flavored inverted-file scheme
- ❑ Intended for web-like collections, e.g the Wikipedia, taking advantage of explicit link information

Other functions

- ❑ A few simple functions that are supported (albeit in another form) in XPath are proposed
 - `eq()`, `gt()`, `lt()`, `le()`, `ge()`
 - `contains()`
- ❑ The parser does not validate the pre-existence of function names
- ❑ Support for wild cards in path expressions (necessary where a DTD is not available)

Path specification

- `node//*` context node or descendant
- `//node*` e.g. `//node6` `//nodename`
- `//*node` e.g. `//mynode` `//this_node`
- `//*node*` e.g. `//mynodeextension`

XOR-RPN : Reverse Polish Notation

- ❑ The parser converts queries from infix to postfix notation.
- ❑ Simplifies the support and evaluation of XOR queries by systems that already support NEXI queries

XOR example

//article[about(.,Germany)

AND NOT

about(.,football)]//sec[about(.,Europe)]

AND

//article[about(., European union
enlargement) AND about(./,German point
of view)]

XOR-RPN

```
//article[about(./.*, German point of view)]  
//article[about(., European union enlargement)]  
AND  
//article//sec[about(., Europe)]  
//article[about(., football)]  
//article[about(., Germany)]  
ANDNOT  
SUPPORT  
AND
```

XOR-RPN

```
//article[about(./.*, German point of view)]  
//article[about(., European union  
enlargement)]  
AND
```

```
//article//sec[about(., Europe)]
```

```
//article[about(., football)]
```

```
//article[about(., Germany)]
```

```
ANDNOT
```

```
SUPPORT
```

```
AND
```

Conclusions

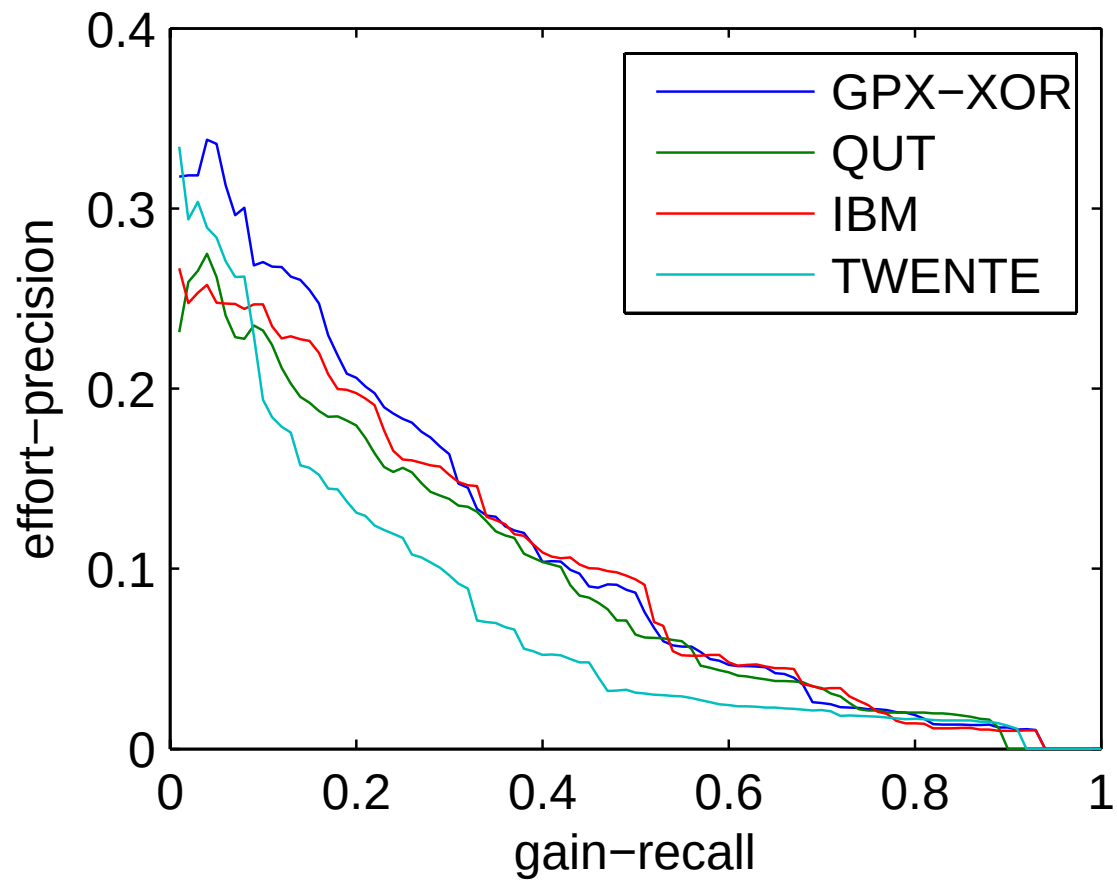
❑ CONCLUSIONS

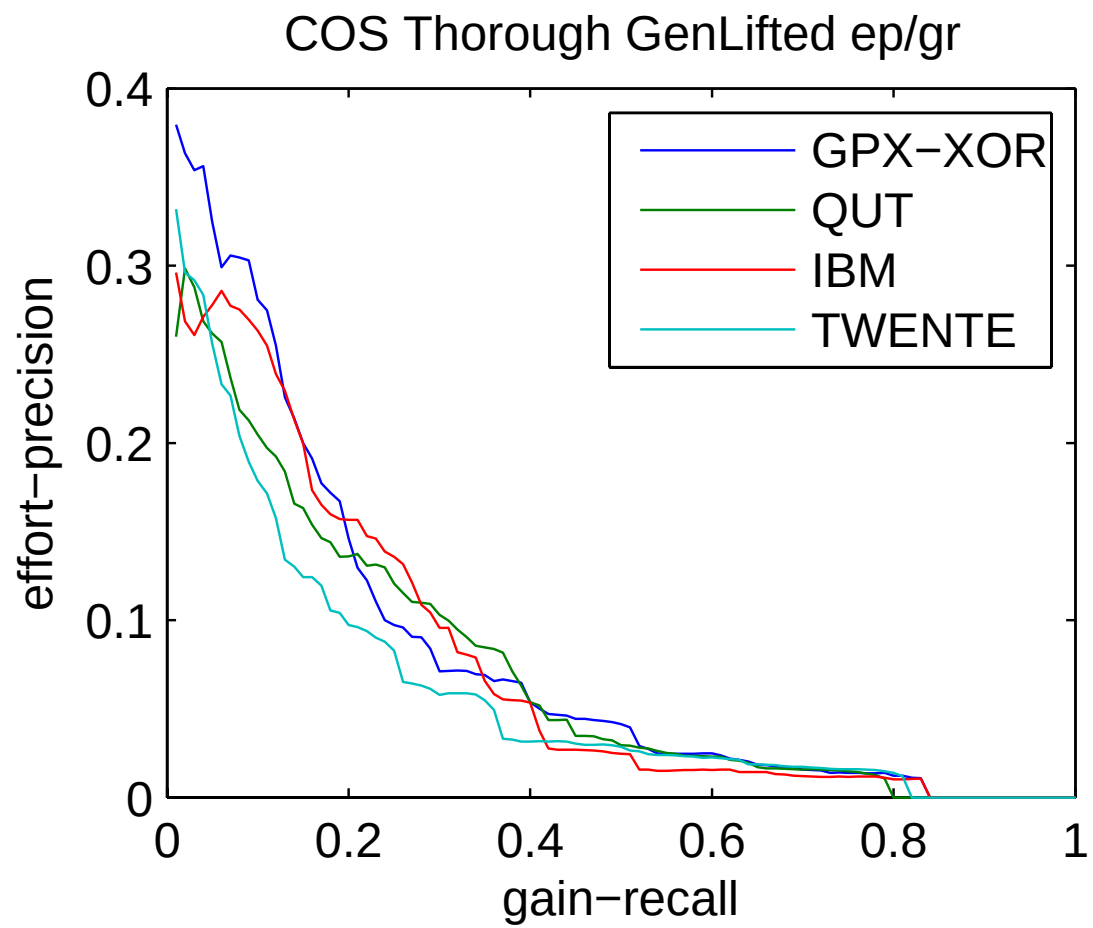
- XOR was explicitly designed to support XML-IR.
- XOR was designed with the experience gained in the INEX natural language queries task, to support more elaborate search but it is useful in general
- more refined control of query content expansion.
- XOR specifies a transformation from infix to postfix notation for easier integration into existing search engines.
- XOR is open ended
- future work will concentrate on providing more functionality in XOR and on open source search engine implementation.





COS Thorough Generalised ep/gr





COS Thorough Strict ep/gr

