XML Information Extraction with XQuery

Processing Wikipedia and Alpino Trees

Gosse Bouma

Information Science
University of Groningen

dept meeting 09/05/08

Outline

XML is everywhere

- Annotated corpora
  - Corea (Coreference), Alpino (Dependency Trees), Imix (Medical Concepts and Relations)
- RSS feeds
- Wikipedia in XML

BBC RSS feed

RSS source

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet title="XSL_formatting" type="text/xsl"
href="/shared/bsp/xsl/rss/nolsol.xsl"?>
<rss xmlns:media="http://search.yahoo.com/mrss/" version="2.0">
  <channel>
    <title>BBC News | Technology | World Edition</title>
    <link>http://news.bbc.co.uk/go/rss/-/2/hi/technology/default.stm</link>
    <description>Visit BBC News for up-to-the-minute news, breaking news, video, audio and feature stories. BBC News pr
    <language>en-gb</language>
    <lastBuildDate>Mon, 24 Sep 2007 08:12:25 GMT</lastBuildDate>

    <item>
      <title>'$100 laptop' to sell to public</title>
      <description>A low-cost laptop, designed for children in developing countries, will go on sale in North America.<
      <link>http://news.bbc.co.uk/go/rss/-/2/hi/technology/6994957.stm</link>
      <guid isPermaLink="false">http://news.bbc.co.uk/1/hi/technology/6994957.stm</guid>
      <pubDate>Mon, 24 Sep 2007 07:39:51 GMT</pubDate>
      <category>Technology</category>
    </item>
```

RSS format

RSS XML format

- RSS feed consists of channels, listing news items
- Elements used in item
  - title, description, link, pubDate, category, ...
- Visualisation in a browser done by (XSLT) stylesheet

RSS

Processing

- Summarize results by Date and Title only
- Sort by date
- Select stories
  - from a given channel,
  - from a given category,
  - on a given topic (keyword match in title)
  - ...

Processing Annotated text

Information Extraction

- Words tagged as body-part, disease, treatment, ...
- Sentences tagged as definition, cause, diagnosis, ..
- Sentences tagged as symptom containing a disease and a disease-symptom

Medical Text with Concept Annotation

Medical Text with Concept Annotation

```
<document id="m1448">
   <con_disease id="7">
    <title level="0">AANDACHTSTEKORTSTOORNIS</title>
   </con_disease>

<section>
  <sec_definition id="107">
    <rel_definition_of id="106">
      <rel_is_symptom_of id="95">
        <con_person id="40">Kinderen</con_person>
         met een
         <con_disease id="14">aandachtstekortstoornis</con_disea
         kunnen zich niet of kort concentreren en....
```

Wikipedia

Wikipedia as XML

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:wx="http://ilps.science.uva.nl/WikiXML/wx"
      xml:lang="nl" lang="nl">
<head>
<title>Albert Speer</title>
</head>
<body>
  <div id="wx_article">
    <wx:section level="1" title="Albert Speer" id="wxsec1">
      <h1 class="pagetitle" id="wx1">Albert Speer</h1>
      <p id="wx6">
        <b id="wx7">Berthold Konrad Hermann Albert Speer</b>
              (<a href="/wiki/Mannheim">Mannheim</a>,
               <a href="/wiki/19_maart">19 maart</a>
               <a href="/wiki/1905">1905</a>
               --
               <a href="/wiki/Londen">Londen</a>,
               <a href="/wiki/1_september">1 september</a>
               <a href="/wiki/1981">1981</a>
             ) was een
               <a href="/wiki/Duits">Duits</a>
               <a href="/wiki/Architect">architect</a>
          die door samenloop van omstandigheden de tweede of derde
          machtigste man van het Derde Rijk werd.
```

Wikipedia

Information extraction

- Dutch version has ± 250K lemma's
- Lemma's contain all kinds of interesting information
  - Link-structure (Lemma for *Speer* refers to *Second World War*)
  - Categories (*Speer* was a *German Architect*)
  - Templates (*Estonia* has *1.4M inhabitants*)

Processing XML

Information Extraction from XML

- RSS: select stories by data, topic, keyword in title, channel, ...
- Medical: find names of diseases, definition sentences, symptom sentences containing a name of a disease, ...
- Wikipedia: All pages in a given category, link-structure, infobox (template) extraction, anchor-texts, cross-language links, ...

Problem

Given a large collection of XML documents

- Find relevant information
- Combine information
- Output results (XML, XHTML, other)

XQuery

- XQuery is a general purpose XML Query Language
- Ideal for extraction tasks
- Simple (not XML) syntax
- Incorporates XPath
- Supported by a range of tools and platforms
  - Saxon (XPath, XSLT, and XQuery processing)
  - XML Databases: eXist, Oracle/Berkeley XML DB, MonetDb, BaseX

Books

```
<?xml version="1.0"?>
<BOOKLIST>
<BOOKS>
  <ITEM CAT="S">
    <TITLE>Number, the Language of Science</TITLE>
    <AUTHOR>Danzig</AUTHOR>
    <PRICE>5.95</PRICE>
  </ITEM>
  <ITEM CAT="F">
    <TITLE>Tales of Grandpa Cat</TITLE>
    <PUBLISHER>Associated Press</PUBLISHER>
    <AUTHOR>Wardlaw, Lee</AUTHOR>
    <PRICE>6.58</PRICE>
  </ITEM>
  <ITEM CAT="S">
    <TITLE>Language &amp;amp; the Science of Number</TITLE>
    <AUTHOR>Danzig</AUTHOR>
    <PRICE>8.95</PRICE>
    <QUANTITY>5</QUANTITY>
  </ITEM>
</BOOKS>
<CATEGORIES DESC="Miscellaneous categories">A list of categories
  <CATEGORY CODE="S" DESC="Science"/>
  <CATEGORY CODE="I" DESC="Science" NOTE="Limited Stock"/>
  <CATEGORY CODE="C" DESC="Computing"/>
  <CATEGORY CODE="X" DESC="Crime"/>
  <CATEGORY CODE="F" DESC="Fiction"/>
  <CATEGORY CODE="U" DESC="Unclassified"/>
</CATEGORIES>
</BOOKLIST>
```

Search Books Catalogue

List all

- books written by Dantzig
- science books
- books with the word Cat in the title
- books sorted by price
- .....

XQuery Example

XQuery

```
for $b in doc('books.xml')/BOOKLIST/BOOKS/ITEM
return
<book>
  { $b/AUTHOR }
  { $b/TITLE }
</book>
```

Result

```
<?xml version="1.0" encoding="UTF-8"?>
<book>
   <AUTHOR>Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides</AUTHOR>
   <TITLE>Design Patterns</TITLE>
</book>
<book>
   <AUTHOR>Bonner</AUTHOR>
   <TITLE>Patterns of Crime in Animal Culture</TITLE>
</book>
```

XQuery Components

Selection

for $VAR in XPath expression selects (a sequence of) elements

- `for $b in doc('books.xml')/BOOKS/ITEM`
- `doc` (and `collection`) are built-in functions for accessing documents

Results

return defines the output

- XML
- Expressions in { curly braces } return the value of that expression

XQuery: Let

Let

- Use let to define variables
- Reuse
- Readability

Example

```
for $b in doc('books.xml')/BOOKLIST/BOOKS/ITEM
let $a := $b/AUTHOR
let $t  := $b/TITLE
return
<book>
  { $a }
  { $t }
</book>
```

XQuery: Where

Where

- where imposes additional restrictions on selection
- Readability

Example

```
for $b in doc('books.xml')/BOOKLIST/BOOKS/ITEM
where $b/@CAT='S'
return
<book>
  { $b/AUTHOR }
  { $b/TITLE  }
</book>
```

Alternative

```
for $b in doc('books.xml')/BOOKLIST/BOOKS/ITEM[@CAT='S']
return
<book>
  { $b/AUTHOR }
  { $b/TITLE  }
</book>
```

XQuery: Order

Order

- order statements produce sorted results
- Order by string value, numerical value, date
- Ascending or descending

Example

```
for $b in doc('books.xml')/BOOKLIST/BOOKS/ITEM
order by $b/AUTHOR
return
<book>
  { $b/AUTHOR }
  { $b/TITLE  }
</book>
```

XQuery: FLWOR expressions

Structure of XQuery statements

- For
- Let
- Where
- Order
- Return

Specifying input documents

for typically defines the input

- Processing a single file
  - `for $doc in doc('project/data.xml')`
  - `for $news in doc('http://www.bbc.uk/rss.xml')`
- Processing multiple files
- Use collection for processing multiple files
  - `for $doc in collection('home/project')`
  - `for $doc in collection('home/project?select=*.xml')`

Specifying input in a separate document

XQuery

```
for $item in
collection('mynews.xml')/rss/channel/item ....
```

mynews.xml

```
<collection>
  <doc href="ad.xml"/>
  <doc href="elsevier.xml"/>
  <doc href="nieuwsnl.xml"/>
  <doc href="http://www.telegraaf.nl/rss/index.xml"/>
  <doc href="http://www.ad.nl/?service=rss"/>
</collection>
```

External Arguments

Passing an argument to a script

```
declare variable $news as xs:string external;

for $item in doc($news)/rss/channel/item
```

```
java net.sf.saxon.Query proc-rss.xq news='bbc.xml'
```

Elements vs data

Select the AUTHOR element

- `for $a in doc('b.xml')/BKS/ITEM/AUTHOR return {$a}`
- `<AUTHOR>Shakespeare</AUTHOR>`

Return only the string

- `for $a in doc('b.xml')/BKS/ITEM/AUTHOR) return <writer>{string($a)}<writer>`
- `<writer>Shakespeare</writer>`

String Processing

XPath Functions

- contains(string,substring)
- subtring-after(string,seperator)
- substring-before(string,seperator)
- tokenize(string,seperator)
- concat(string,string,string*)
- string-join(string,seperator)

Reformatting dates

RSS publication dates do not follow XSchema date type

25 09 2007 09:36:40 → 2007-09-25T09:36:40

XQuery

```
let $date0 := tokenize(string($item/pubDate), " ")
let $date := concat($date0[4],'-',$date0[3],'-',
                    $date0[2],'T',$date0[5])
```

Producing HTML output

```
<html>
  <head>
    <title>A list of books</title>
  </head>
  <body>
    <h1>A list of books</h1>
    <p>Here are some interesting books:</p>
    <ul> {
for $b in //BOOKS/ITEM
order by $b/TITLE return
<li><i>{string($b/TITLE)}</i> by {string($b/AUTHOR)}
</li>
    } </ul>
  </body>
</html>
```

Regular Expressions

matches(string,pattern)

```
let $month := "(januari|februari|...|december)"
let $year  := "^[12][0-9][0-9][0-9]"

let $para  := $doc/body/p[1]

let $bornmonth  :=
    string($para/a[matches(@title,$month)][1])
let $bornyear   :=
     string($para/a[matches(@title,$year)][1])

return
<born person='{string($doc/head/title)}'>
{$bornmonth, $bornyear}
</born>
```

Person Facts (33K borndate facts)

```
<wikipage compact-file-id="p00nnnnn/1.xml">
   <title>Albert Speer</title>
   <fullname>Berthold Konrad Hermann Albert Speer</fullname>
   <borndate>19 maart 1905</borndate>
   <bornloc>Mannheim</bornloc>
   <dieddate>1 september 1981</dieddate>
   <diedloc>Londen</diedloc>
</wikipage>
<wikipage compact-file-id="p00nnnnn/6.xml">
   <title>Anthony Fokker</title>
   <fullname>Anton Herman Gerard Fokker</fullname>
   <borndate>6 april 1890</borndate>
   <dieddate>23 december 1939</dieddate>
</wikipage>
<wikipage compact-file-id="p00nnnnn/7.xml">
   <title>Albert Plesman</title>
   <fullname>Albert Plesman</fullname>
   <borndate>7 september 1889</borndate>
   <bornloc>Den Haag</bornloc>
   <dieddate>31 december 1953</dieddate>
</wikipage>
....
```

Sorting by String Value

XQuery

```
for $b in doc('books.xml')/BOOKLIST/BOOKS/ITEM
order by $b/PRICE
return
<book>
  { $b/AUTHOR }
  { $b/TITLE }
  { $b/PRICE }
</book>
```

Result

```
<book>
   <AUTHOR>Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides</AUTHOR>
   <TITLE>Design Patterns</TITLE>
   <PRICE>49.95</PRICE>
</book>
<book>
   <AUTHOR>Danzig</AUTHOR>
   <TITLE>Number, the Language of Science</TITLE>
   <PRICE>5.95</PRICE>
```

Sorting by Number

XQuery

```
for $b in doc('books.xml')/BOOKLIST/BOOKS/ITEM
order by number($b/PRICE)
return
<book>
  { $b/AUTHOR }
  { $b/TITLE }
  { $b/PRICE }
</book>
```

Result

```
<book>
   <AUTHOR>Stephen R. Davis</AUTHOR>
   <TITLE>Learn Java Now</TITLE>
   <PRICE>9.95</PRICE>
</book>
<book>
   <AUTHOR>Milne, A. A.</AUTHOR>
   <TITLE>When We Were Very Young</TITLE>
   <PRICE>12.50</PRICE>
</book>
```

Reverse Sorting

XQuery

```
for $b in doc('books.xml')/BOOKLIST/BOOKS/ITEM
order by number($b/PRICE) descending
return
<book>
  { $b/AUTHOR }
  { $b/TITLE }
  { $b/PRICE }
</book>
```

Result

```
<book>
   <AUTHOR>Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides</AUTHOR>
   <TITLE>Design Patterns</TITLE>
   <PRICE>49.95</PRICE>
</book>
<book>
   <AUTHOR>Bonner</AUTHOR>
   <TITLE>Patterns of Crime in Animal Culture</TITLE>
   <PRICE>15.95</PRICE>
</book>
```

Nesting

Valid XML requires a root element

```
<listofbooks>
{ for $b in doc('books.xml')/BOOKLIST/BOOKS/ITEM
  order by number($b/PRICE) descending
  return
  <book>
    { $b/TITLE }
    { $b/PRICE }
  </book>
}
</listofbooks>
```

Conditional Statements

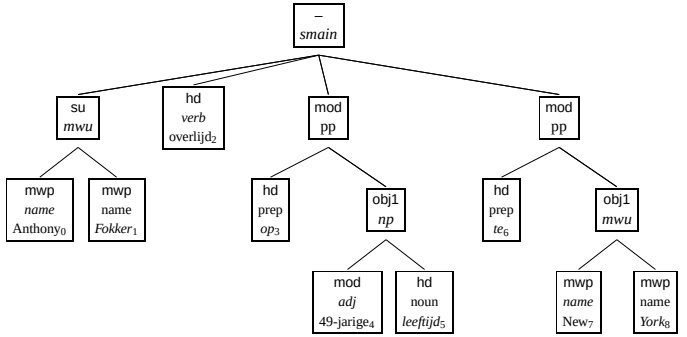
if-then-else

```
for $b in doc('books.xml')/BOOKLIST/BOOKS/ITEM
let $price := if (number($b/PRICE) gt 10.00)
              then 'expensive'
              else 'bargain!'
return
<book>
  { $b/TITLE }
  <price>{$price}</price>
 </book>
```

Nested

```
for $b in doc('books.xml')/BOOKLIST/BOOKS/ITEM
return
<book>
  { $b/TITLE }
  <price>{if (number($b/PRICE) gt 10.00)
          then 'expensive'
          else 'bargain!'
        }</price>
</book>
```

Alpino Dependency Trees



Anthony Fokker overleed op 49-jarige leeftijd te New York
Anthony Fokker died at age 49 in New York

Searching Treebanks

- Automatically Parsed Treebanks Widely Used
- Tools for Search and Extraction
  - Treebank-specific
  - Application-specific
- Most treebanks in XML
  - Use common standards and tools
  - XQuery intended for XML search and extraction
  - Module system supports reuse, hides complexity

Automatically Annotated Corpora

Corpus	Size (M words)	Genre
Twente News Corpus	500	newspapers
CLEF	80	newspapers
Wikipedia	50	web encyclopedia
Europarl	28	proceedings
Medical	3	handbooks, web

Alpino Parser Accuracy

suite	type	sentences	tokens/sent	f-score dependencies
alpino	news-paper	7,136	20	88.5%
trouw	news-paper	1,400	17	91.1%
clef	questions	1,346	8	96.3%
CGN	spoken	130,000	9	71.0%

Applications

- Manually (partially) corrected Treebanks
  - 

Alpino treebank (140.000 words)
  - 
D-COI (200.000 words)
  - 

LASSY (500-800M words)
- Syntactic Disambiguation
  - Bilexical preferences (van Noord, IWPT 2007)
- Acquisition of Lexical Knowledge
  - Terms, Definitions, Similar words, Hypernyms, ...

Applications

- 

Question Answering
  - 
CLEF 05, 06, 07
- Corpus Linguistics
  - Dative Alternation, Support Verb constructions, Focus Particles, Emphatic Reflexives, ...

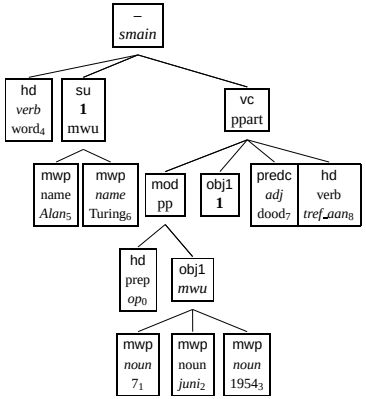
Frequency of Causative Alternation Verbs

- Requires parsed corpus
  - Subcategorization-frame used must be identified
- Ignore verbs which allow both Object Drop and Causative alternation
  - Hij kookt de aardappelen (*He cooks the potatoes*)
  - De aardappelen koken (*The potatoes are cooking*)
  - Hij kookt regelmatig (*He cooks regularly*)
- Various non-finite intransitive patterns are ambiguous
  - Het ijs is gesmolten
    - *The ice is/has melted* (passive/perfect)
  - Hij laat de suiker smelten
    - *He has someone melt the sugar*
    - *He lets the sugar melt*

Causative Alternation in TwNC (500M words)

Verb		Trans	%	Intrans	%
verkleinen	*to diminish*	1.067	93	81	7
vergroten	*to increase*	3.692	93	273	7
oplossen	*to solve*	3.878	81	884	19
verbeteren	*to improve*	2.852	64	1.613	36
breken	*to break*	6.246	61	4.044	39
opwarmen	*to heat up*	215	60	142	40
verbranden	*burn*	660	57	506	43
smelten	*to melt*	381	34	734	66
ontdooien	*to defrost*	66	29	163	71
veranderen	*to change*	4.219	27	11.411	73
afkoelen	*to cool down*	96	19	402	81
verdrink	*to drown*	171	11	1.373	89

Alpino Dependency Trees



Op 7 juni 1954 werd Alan Turing dood aangetroffen
On June 7, 1954, Alan Turing was found dead

Alpino Dependency Trees

```
<alpino_ds version=''1.1.''>
<node cat="smain" rel="--">
  <node pos="verb" rel="hd" root="word"/>
  <node cat="mwu" end="7" index="1" rel="su">
    <node pos="name" rel="mwp" root="Alan" />
    <node pos="name" rel="mwp" root="Turing"/>
  </node>
  <node cat="ppart" rel="vc">
    <node cat="pp" rel="mod">
      ...
      </node>
    </node>
    <node index="1" rel="obj1"/>
    <node pos="adj" rel="predc" root="dood"/>
    <node pos="verb" rel="hd" root="tref_aan"/>
  </node>
</node>
```

IE from Alpino XML

- Alpino XML was designed to be maximally simple and compact
- Finding linguistic elements can be surprisingly difficult
  - NPs can be `pos=noun`, `pos=name`, `pos=pron`, `cat=NP`, `cat=mwu`, `cat=conj`, ...
  - Content of an NP can be a phrase or just an index
- Some issues need occur in many applications
  - Finding the yield of a phrase
  - Finding the head of a phrase
  - Resolving indices
  - Finding date expressions, location expressions, person names, ...
  - ....

Common Tasks

- Tasks
  - Search trees (graphs)
  - Extract specific information
- Characteristics
  - Queries become quite complex
  - Overlap between queries

Current Situation

- Alpino Treebank uses an XML data format
  - Compact
  - Syntactic parent/child relations $\Leftrightarrow$ parent/child relations of XML elements
- Search Tool
  - dtsearch (based on XPath)
- Extraction Tools
  - XSLT
  - General programming language (Prolog/Perl/Java/Python) with XML support

XPath 2.0

- General purpose search language for XML documents
- Supported by a range of tools and libraries
- Boolean search

```
node[@cat='np' and not(@rel='su' or @rel='obj1')]
```

- Search for child/parent, descendant/ancestor, sibling

```
//node[@cat='np']/
     node[@rel='hd' and ../node[@rel='det']]
```

- Regular expressions

```
//node[matches(string(@root),'^[A..Z]')]
```

Frequently Asked Questions

- Match all Proper Names

```
//node[@pos='name' or
       (@pos='mwu' and node[@pos='name'] ) ]
```

- Match all finite verbs

```
//node[@cat='smain' or @cat='sv1' or @cat='ssub']/
     node[@rel='hd']


//node[@pos='verb' and
    (@infl='sg' or @infl='sg1' or @infl='sg3' or
     @infl='pl' or @infl='past(sg)' ...) ]
```

- Match all NPs....

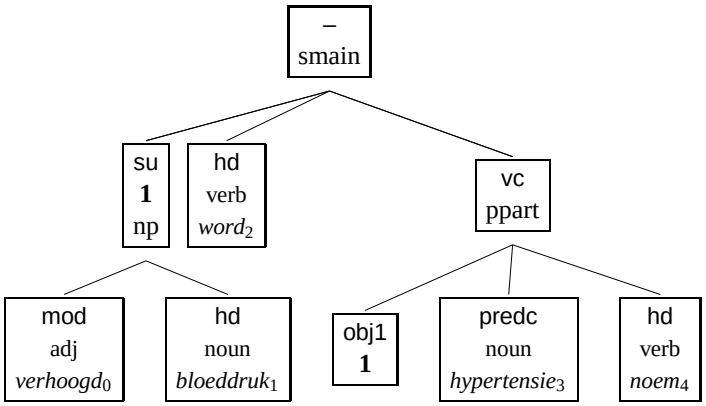
Frequently Asked Questions

NPs (markables for coreference annotation....)

```
//node[    not(    @rel="mwp"
                 or  @rel="rhd"
                 or (@rel="hd" and not(@cat="mwu"))
                 or (@rel="app" and (@pos="name" or (@cat="mwu" and
node[@pos="name"])))
                 or (@rel="det" and @cat="np")
                )
        and ( @cat="np"
           or @pos="pron"
           or @pos="name"
           or @pos="noun"
           or (@cat="mwu"  and node[@pos="name" or @pos="noun"] and
not(@rel="hd"))
           or (@cat="conj" and node[@cat="np" or @pos="noun" or @pos="name" or
                                    @pos="pron" or (@cat="mwu" and
node[@pos="name"])
                                   ])
           or (@pos="det" and (@root ="mijn" or @root="jouw" or @root="je"
                               or @root="zijn" or @root="haar" or @root="ons" or
@root="onze"
                               or @root="jullie" or @root="uw" or @root="hun"))
           or (@pos="num" and not(@rel="det" or @rel="mod"))
           or (@pos="det" and not(@rel="det" or @rel="cnj"))
           )
        ]
```

Frequently Asked Questions

- Alpino treebanks contain dependency graphs



High blood pressure is called hypertension

Extracting arguments of a verb

- Extract the root of NPs occurring as object of a given verb
- Base case

```
//node[node[@root='noem']]/
    node[@rel='obj1' and @cat='np']/
                  node[@rel='hd']/@root
```

- Root of NP co-indexed with an object of a given verb

```
//node[@cat='np' and
       @index=ancestor::alpino_ds/descendant::node
                 [@rel='obj1' and
                  ../node[@root='noem'] ]/@index ]/
   node[@rel='hd']/@root
```

XQuery Example (Verb-Obj pairs)

```
for $node in collection('Trees')/alpino_ds//node

let $verb := $node/../node[@rel='hd' and @pos='verb']

where $node[@cat='np' and @rel='obj1']

order by base-uri($node)

return
<obj-verb-pair
      obj='{string($node/node[@rel='hd']/@root)}'
      verb='{string($verb/@root)}'
      file='{base-uri($node)}'     />
```

Finding Location Names

```
for $node in
    collection('capital-trees')//node[@pos='name'
                                      and @neclass='LOC']

return
text{$node/@word, "&#10;"}
```

```
Belfast
Verenigd
Koninkrijk
Brasília
Rio
de
Janeiro
Kopenhagen
```

Multi word units

```
<node cat="mwu" rel="app">
  <node neclass="LOC" pos="name" rel="mwp" word="Rio"/>
  <node neclass="LOC" pos="name" rel="mwp" word="de"/>
  <node neclass="LOC" pos="name" rel="mwp" word="Janeiro"/>
</node>
```

Finding Location Names 2

```
for $node in collection('capital-trees')//node[
            (not(@rel='mwp') and @neclass='LOC') or
            (@cat='mwu' and node[@neclass='LOC'])
           ]
let $words :=
    if ( $node/@word ) then
         string($node/@word)
    else string-join($node/node/@word,' ')
return
text{$words,"&#10;"}
```

```
Samarra
Brussel
Tsjechische Republiek
Minas Gerais
Rio de Janeiro
Oost-Duitsland
DDR
Oost-Berlijn
```

Words function

```
declare function local:words($node as element(node)) as xs:strin
  { if ( $node/@word ) then
         string($node/@word)
    else string-join($node/node/@word,' ')
  } ;

for $node in collection('capital-trees')//node[
            (not(@rel='mwp') and @neclass='LOC') or
            (@cat='mwu' and node[@neclass='LOC'])
           ]

return
text{local:words($node),"&#10;"}
```

Alpino Module

```
module namespace alpino="alpino.xq" ;

declare function alpino:yield($constituent as element(node))
                                                    as xs:string
{ let $words :=
    for $leaf in $constituent/descendant-or-self::node[@word]
    order by number($leaf/@begin)
    return $leaf/@word
  return string-join($words," ")
} ;
```

Using a Module

```
import module namespace alpino = "alpino.xq" at "alpino.xq" ;

for $node in collection('capital-trees')//node[
            (not(@rel='mwp') and @neclass='LOC') or
            (@cat='mwu' and node[@neclass='LOC'])
           ]

return
text{alpino:yield($node),"&#10;"}
```

Schema Types

```
declare function
          alpino:date-node($constituent as element(node))
                                                  as xs:boolean
{ if  (  $constituent[@special="tmp"]
      or $constituent[@cat = "mwu" and node[@special="tmp"]]
      )  then  true()
  else if ( $constituent[@cat = "pp" and node[@rel="obj1"]] )
       then alpino:date-node($constituent/node[@rel="obj1"])
       else false()
};

declare function
    alpino:date-dependents($head as element(node))
                                          as element(node)*
 { for $dep in $head/../node
   where alpino:date-node($dep)
   return $dep    };
```

Alpino XQuery Module

- **resolve-index** Return
  - co-indexed node for index nodes, otherwise
  - self
- **head-of** Return the (first) lexical head of an (index) node
- **yield** Return sorted concatenation of all the `@word` values in descendants of a node
- **root-string** Return
  - the value of `@root` if defined,
  - concatenation of `node/@root` for`@cat='mwu'` nodes,
  - prefix heads with `@root` of `@pos='fixed'` sisters

Alpino XQuery Module

- **person-node**
  - `node[@neclass='PER']` or
  - `node[@cat='mwu]/node[@neclass='PER']`
- **semantic-role** Value of
  - `@rel` or
  - `@rel` of the mother (for heads, appositions, and conjuncts)
- **selector-of** Sister with `@rel='hd'` of
  - self or
  - the mother (for head, appositions, and conjuncts)

Using Alpino XQuery Module

- Scripts for specific tasks can be simplified substantially
  - Extracting object-verb triples
  - XSLT: 149 lines
  - XQuery: 28 lines
- Avoids development of corpus-specific tools
  - Public standard: Lots of documentation and software tools available
  - Writing basic XQuery scripts is not hard
  - Real XML is complex
  - Complexity is dealt with in a corpus-specific module
- Scales to large corpora
  - XML database solutions

More on XQuery

Tutorials and Documentation

- Tutorials
  - Walmsley, Introduction to XQuery
  - Stylusstudio tutorials
- Walmsley, XQuery (O 'Reilly)
- Documentation
  - `http://www.w3.org/TR/xquery/`
  - `http://www.w3.org/TR/xpath20/`
  - `http://www.w3.org/TR/xpath-functions/`

Saxon

- Saxon (`www.saxonica.com`) is a XSLT and XQuery parser
- Implemented in Java
- Installed on skuld at `/users1/gosse/src/saxon8-8`
- `java -cp /storage/gosse/src/saxon8-8/saxon8.jar net.sf.saxon.Query example.xq`

Saxon and dictzip files

- Alpino and Wikipedia XML stored in dictzip archives
- Cannot be accessed directly by Saxon
- Solution
  - Start a treebank-server
  - Run xqclient to run an xquery-script on a list of files

```
$ export WIKI=/storage/geertk/uva-wiki-corpus/compact
$ treebank-server -l -c $WIKI &
  (starts the server)
$ dtlist -r  $WIKI | xqclient -s person-facts.xq
  (applies person-facts.xq to all files in WIKI archive
```

- Efficiency:
  - 2.5 hrs: Extracting page-title/anchor text from hypertext links in the English Wikipedia (Result: 3.5 GB)
  - 15 hrs: Extracting all triples for computing distributional similarity from TwNC (Result: 1.2 GB)

XML Databases supporting XQuery

XML Databases

- eXist
- MonetDb
- BaseX

Experiences

- MonetDb is very picky about XQuery
- BaseX slower than Saxon on some (most) tasks
- eXist seems promising, but not tested on really large datasets

Conclusions

- XQuery ideal for Information Extractrion from XML
- Modules support working with complex data
- Scales to large data-sets

Conclusions

- Syntactically annotated corpora
  - useful for a range of applications
  - several (non-trivial) common tasks
- Search and Extraction
  - can be supported using XPath, XQuery, and related XML standards