Representing and Querying Multi-dimensional Markup for Question Answering

Wouter Alink, Valentin Jijkoun, David Ahn, Maarten de Rijke
ISLA, University of Amsterdam
alink,jijkoun,ahn,mdr@science.uva.nl

Peter Boncz, Arjen de Vries
CWI, Amsterdam, The Netherlands
boncz,arjen@cwi.nl

Abstract

This paper describes our approach to representing and querying multi-dimensional, possibly overlapping text annotations, as used in our question answering (QA) system. We use a system extending XQuery, the W3C-standard XML query language, with new axes that allow one to jump easily between different annotations of the same data. The new axes are formulated in terms of (partial) overlap and containment. All annotations are made using stand-off XML in a single document, which can be efficiently queried using the XQuery extension. The system is scalable to gigabytes of XML annotations. We show examples of the system in QA scenarios.

1 Introduction

Corpus-based question answering is a complex task that draws from information retrieval, information extraction and computational linguistics to pinpoint information users are interested in. The flexibility of natural language means that potential answers to questions may be phrased in different ways—lexical and syntactic variation, ambiguity, polysemy, and anaphoricity all contribute to a gap between questions and answers. Typically, QA systems rely on a range of linguistic analyses, provided by a variety of different tools, to bridge this gap from questions to possible answers.

In our work, we focus on how we can integrate the analyses provided by completely independent linguistic processing components into a uniform QA framework. On the one hand, we would like to be able, as much as possible, to make use of off-the-shelf NLP tools from various sources without having to worry about whether the output of the tools are compatible, either in a strong sense of forming a single hierarchy or even in a weaker sense of simply sharing common tokenization. On the other hand, we would like to be able to issue simple and clear queries that jointly draw upon annotations provided by different tools.

To this end, we store annotated data as stand-off XML and query it using an extension of XQuery with our new StandOff axes, inspired by (Burkowski, 1992). Key to our approach is the use of stand-off annotation at every stage of the annotation process. The source text, or character data, is stored in a Binary Large OBject (BLOB), and all annotations, in a single XML document. To generate and manage the annotations we have adopted XIRAF (Alink, 2005), a framework for integrating annotation tools which has already been successfully used in digital forensic investigations.

Before performing any linguistic analysis, the source documents, which may contain XML metadata, are split into a BLOB and an XML document, and the XML document is used as the initial annotation. Various linguistic analysis tools are run over the data, such as a named-entity tagger, a temporal expression (timex) tagger, and syntactic phrase structure and dependency parsers. The XML document will grow during this analysis phase as new annotations are added by the NLP tools, while the BLOB remains intact. In the end, the result is a fully annotated stand-off document, and this annotated document is the basis for our QA system, which uses XQuery extended with the new axes to access the annotations.

The remainder of the paper is organized as follows. In Section 2 we briefly discuss related work. Section 3 is devoted to the issue of querying multi-dimensional markup. Then we describe how we coordinate the process of text annotation, in Sec-

tion 4, before describing the application of our multi-dimensional approach to linguistic annotation to question answering in Section 5. We conclude in Section 6.

2 Related Work

XML is a tree structured language and provides very limited capabilities for representing several annotations of the same data simultaneously, even when each of the annotations is tree-like. In particular, in the case of inline markup, multiple annotation trees can be put together in a single XML document only if elements from different annotations do not cross each other's boundaries.

Several proposals have tried to circumvent this problem in various ways. Some approaches are based on splitting overlapping elements into fragments. Some use SGML with the CONCUR feature or even entirely different markup schemes (such as LMNL, the Layered Markup and Annotation Language (Piez, 2004), or GODDAGs, generalized ordered-descendant directed acyclic graphs (Sperberg-McQueen and Huitfeldt, 2000)) that allow arbitrary intersections of elements from different hierarchies. Some approaches use empty XML elements (*milestones*) to mark beginnings and ends of problematic elements. We refer to (DeRose, 2004) for an in-depth overview.

Although many approaches solve the problem of *representing* possibly overlapping annotations, they often do not address the issue of accessing or *querying* the resulting representations. This is a serious disadvantage, since standard query languages, such as XPath and XQuery, and standard query evaluation engines cannot be used with these representations directly.

The approach of (Sperberg-McQueen and Huitfeldt, 2000) uses GODDAGs as a conceptual model of multiple tree-like annotations of the same data. Operationalizing this approach, (Dekhtyar and Iacob, 2005) describes a system that uses multiple inline XML annotations of the same text to build a GODDAG structure, which can be queried using EXPath, an extension of XPath with new axis steps.

Our approach differs from that of Dekhtyar and Iacob in several ways. First of all, we do not use multiple separate documents; instead, all annotation layers are woven into a single XML document. Secondly, we use stand-off rather than inline annotation; each character in the original document is referred to by a unique offset, which means that specific regions in a document can be denoted unambiguously with only a start and an end offset. On the query side, our extended XPath axes are similar to the axes of Dekhtyar and Iacob, but less specific: e.g., we do not distinguish between left-overlapping and right-overlapping character regions.

In the setting of question answering there are a few examples of querying and retrieving semistructured data. Litowski (Litkowksi, 2003; Litkowksi, 2004) has been advocating the use of an XML-based infrastructure for question answering, with XPath-based querying at the back-end, for a number of years. Ogilvie (2004) outlines the possibility of using multi-dimensional markup for question answering, with no system or experimental results yet. Jijkoun et al. (2005) describe initial experiments with XQuesta, a question answering system based on multi-dimensional markup.

3 Querying Multi-dimensional Markup

Our approach to markup is based on stand-off XML. Stand-off XML is already widely used, although it is often not recognized as such. It can be found in many present-day applications, especially where annotations of audio or video are concerned. Furthermore, many existing multi-dimensional-markup languages, such as LMNL, can be translated into stand-off XML.

We split annotated data into two parts: the *BLOB* (Binary Large OBject) and the XML annotations that refer to specific regions of the BLOB. A BLOB may be an arbitrary byte string (e.g., the contents of a hard drive (Alink, 2005)), and the annotations may refer to regions using positions such as byte offsets, word offsets, points in time or frame numbers (e.g., for audio or video applications). In text-based applications, such as described in this paper, we use character offsets. The advantage of such character-based references over word- or token-based ones is that it allows us to reconcile possibly different tokenizations by different text analysis tools (cf. Section 4).

In short, a multi-dimensional document consists of a BLOB and a set of stand-off XML annotations of the BLOB. Our approach to querying such documents extends the common XML query languages XPath and XQuery by defining 4 new axes that allow one to move from one XML tree to another. Until recently, there have been very few

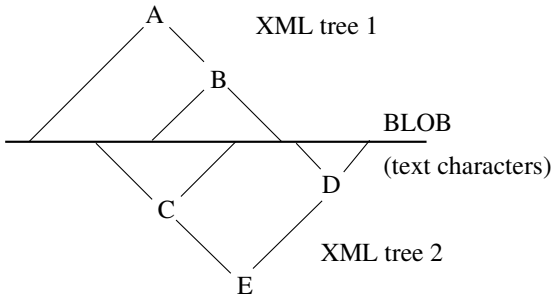

Figure 1: Two annotations of the same data.

approaches to querying stand-off documents. We take the approach of (Alink, 2005), which allows the user to relate different annotations using containment and overlap conditions. This is done using the new StandOff XPath axis steps that we add to the XQuery language. This approach seems to be quite general: in (Alink, 2005) it is shown that many of the query scenarios given in (Iacob et al., 2004) can be easily handled by using these StandOff axis steps.

Let us explain the axis steps by means of an example. Figure 1 shows two annotations of the same character string (BLOB), where the first XML annotation is

```
<A start="10" end="50">
  <B start="30" end="50"/>
</A>
```

and the second is

```
<E start="20" end="60">
  <C start="20" end="40"/>
  <D start="55" end="60">
</E>
```

While each annotation forms a valid XML tree and can be queried using standard XML query languages, together they make up a more complex structure.

StandOff axis steps, inspired by (Burkowski, 1992), allow for querying overlap and containment of regions, but otherwise behave like regular XPath steps, such as `child` (the step between `A` and `B` in Figure 1) or `sibling` (the step between `C` and `D`). The new StandOff axes, denoted with `select-narrow`, `select-wide`, `reject-narrow`, and `reject-wide` select contained, overlapping, non-contained and non-overlapping region elements, respectively, from possibly distinct layers of XML annotation of the data. Table 1 lists some examples for the annotations of our example document.

In XPath, the new axis steps are used in exactly the same way as the standard ones. For example,

Context	Axis	Result nodes
`A`	select-narrow	`B C`
`A`	select-wide	`B C E`
`A`	reject-narrow	`E D`
`A`	reject-wide	`D`

Table 1: Example annotations.

the XPath query:

```
//B/select-wide::*
```

returns all nodes that overlap with the span of a `B` node: in our case the nodes `A`, `B`, `C` and `E`. The query:

```
//*[./select-narrow::B]
```

returns nodes that contain the span of `B`: in our case, the nodes `A` and `E`.

In implementing the new steps, one of our design decisions was to put all stand-off annotations in a single document. For this, an XML processor is needed that is capable of handling large amounts of XML. We have decided to use MonetDB/XQuery, an XQuery implementation that consists of the Pathfinder compiler, which translates XQuery statements into a relational algebra, and the relational database MonetDB (Grust, 2002; Boncz, 2002).

The implementation of the new axis steps in MonetDB/XQuery is quite efficient. When the XMark benchmark documents (XMark, 2006) are represented using stand-off notation, querying with the StandOff axis steps is interactive for document size up to 1GB. Even millions of regions are handled efficiently. The reason for the speed of the StandOff axis steps is twofold. First, they are accelerated by keeping a database index on the region attributes, which allows fast merge-algorithms to be used in their evaluation. Such merge-algorithms make a single linear scan through the index to compute each StandOff step. The second technical innovation is "loop-lifting." This is a general principle in MonetDB/XQuery(Boncz et al., 2005) for the efficient execution of XPath steps that occur *nested* in XQuery iterations (i.e., inside `for`-loops). A naive strategy would invoke the StandOff algorithm for each iteration, leading to repeated (potentially many) sequential scans. Loop-lifted versions of the StandOff algorithms, in contrast, handle all iterations together in one sequential scan, keeping the average complexity of the StandOff steps linear.

The StandOff axis steps are part of release 0.10 of the open-source MonetDB/XQuery product, which can be downloaded from `http://www.monetdb.nl/XQuery`.

In addition to the StandOff axis steps, a keyword search function has been added to the XQuery system to allow queries asking for regions containing specific words. This function is called `so-contains($node, $needle)` which will return a boolean specifying whether `$needle` occurs in the given region represented by the element `$node`.

4 Combining Annotations

In our QA application of multi-dimensional markup, we work with corpora of newspaper articles, each of which comes with some basic annotation, such as title, body, keywords, timestamp, topic, etc. We take this initial annotation structure and split it into raw data, which comprises all textual content, and the XML markup. The raw data is the BLOB, and the XML annotations are converted to stand-off format. To each XML element originally containing textual data (now stored in the BLOB), we add a `start` and `end` attribute denoting its position in the BLOB.

We use a separate system, XIRAF, to coordinate the process of automatically annotating the text. XIRAF (Figure 2) combines multiple text processing tools, each having an input descriptor and a tool-specific wrapper that converts the tool output into stand-off XML annotation. Figure 3 shows the interaction of XIRAF with an automatic annotation tool using a wrapper.

The input descriptor associated with a tool is used to select regions in the data that are candidates for processing by that tool. The descriptor may select regions on the basis of the original metadata or annotations added by other tools. For example, both our sentence splitter and our temporal expression tagger use original document metadata to select their input: both select document text, with `//TEXT`. Other tools, such as syntactic parsers and named-entity taggers, require separated sentences as input and thus use the output annotations of the sentence splitter, with the input descriptor `//sentence`. In general, there may be arbitrary dependencies between text-processing tools, which XIRAF takes into account.

In order to add the new annotations generated by a tool to the original document, the output of the tool must be represented using stand-off XML annotation of the input data. Many text processing tools (e.g., parsers or part-of-speech taggers) do not produce XML annotation per se, but their output can be easily converted to stand-off XML annotation. More problematically, text processing tools may actually modify the input text in the course of adding annotations, so that the offsets referenced in the new annotations do not correspond to the original BLOB. Tools make a variety of modifications to their input text: some perform their own tokenization (i.e., inserting whitespaces or other word separators), silently skip parts of the input (e.g., syntactic parsers, when the parsing fails), or replace special symbols (e.g., parentheses with `-LRB-` and `-RRB-`). For many of the available text processing tools, such possible modifications are not fully documented.

XIRAF, then, must map the output of a processing tool back to the original BLOB before adding the new annotations to the original document. This re-alignment of the output of the processing tools with the original BLOB is one of the major hurdles in the development of our system. We approach the problems systematically. We compare the text data in the output of a given tool with the data that was given to it as input and re-align input and output offsets of markup elements using an edit-distance algorithm with heuristically chosen weights of character edits. After re-aligning the output with the original BLOB and adjusting the offsets accordingly, the actual data returned by the tool is discarded and only the stand-off markup is added to the existing document annotations.

5 Question Answering

XQuesta, our corpus-based question-answering system for English and Dutch, makes use of the multi-dimensional approach to linguistic annotation embodied in XIRAF. The system analyzes an incoming question to determine the required answer type and keyword queries for retrieving relevant snippets from the corpus. From these snippets, candidate answers are extracted, ranked, and returned.

The system consults Dutch and English newspaper corpora. Using XIRAF, we annotate the corpora with named entities (including type information), temporal expressions (normalized to ISO values), syntactic chunks, and syntactic parses (dependency parses for Dutch and phrase structure

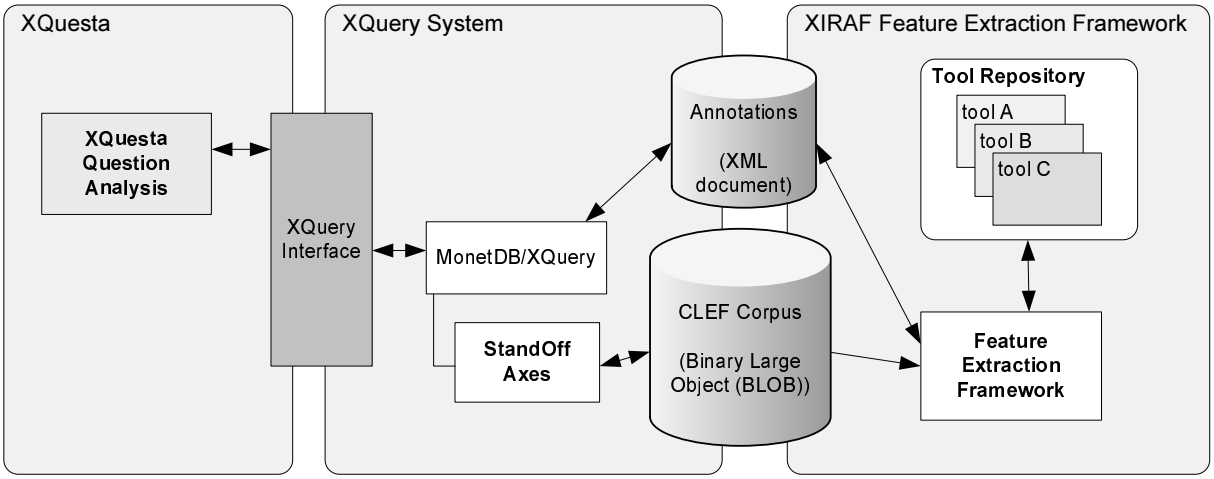

Figure 2: XIRAF Architecture

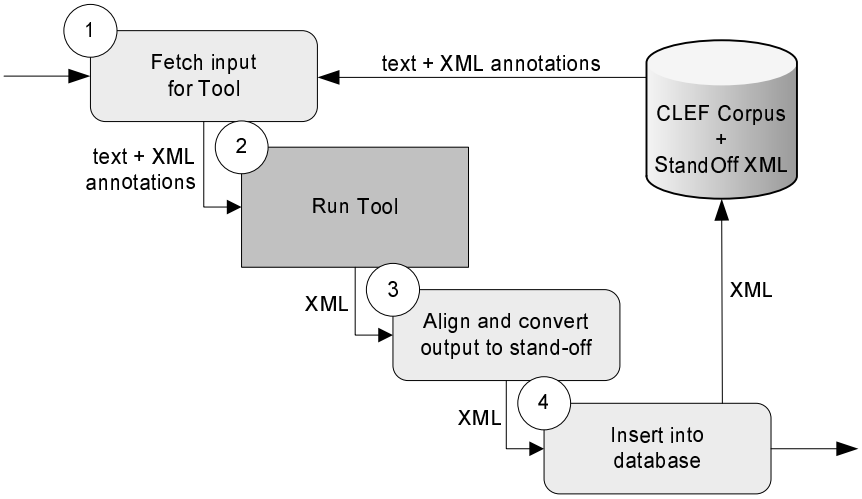

Figure 3: Tool Wrapping Example

parses for English).

XQuesta's question analysis module maps questions to both a keyword query for retrieval of relevant passages and a query for extracting candidate answers. For example, for the question *How many seats does a Boeing 747 have?*, the keyword query is *Boeing 747 seats*, while the extraction query is the pure XPath expression:

```
//phrase[@type="NP"][.//WORD
[@pos="CD"]][so-contains(.,
"seat")]
```

This query can be glossed: find `phrase` elements of type NP that dominate a `word` element tagged as a cardinal determiner and that also contain the string "seat". Note that `phrase` and `word` elements are annotations generated by a single tool (the phrase-structure parser) and thus in the same annotation layer, which is why standard XPath can be used to express this query.

For the question *When was Kennedy assassinated?*, on the other hand, the extraction query is an XPath expression that uses a StandOff axis:

```
//phrase[@type="S" and headword=
"assassinated" and so-contains(.,
"Kennedy")]/select-narrow::timex
```

This query can be glossed: find temporal expressions whose textual extent is contained inside a sentence (or clause) that is headed by *assassinated* and contains the string "Kennedy". Note that `phrase` and `timex` elements are generated by different tools (the phrase-structure parser and the temporal expression tagger, respectively), and therefore belong to different annotation layers. Thus, the `select-narrow::` axis step must be used in place of the standard `child::` or `descendant::` steps.

As another example of the use of the StandOff axes, consider the question *Who killed John*

F. Kennedy?. Here, the keyword query is *kill John Kennedy*, and the extraction query is the following (extended) XPath expression:

```
//phrase[@type="S" and headword=
"killed" and so-contains(.,
 "Kennedy")]/phrase[@type="NP"]/
select-wide::ne[@type="per"]
```

This query can be glossed: find person named-entities whose textual extent overlaps the textual extent of an NP phrase that is the subject of a sentence phrase that is headed by *killed* and contains the string "Kennedy". Again, `phrase` elements and `ne` elements are generated by different tools (the phrase-structure parser and named-entity tagger, respectively), and therefore belong to different annotation layers. In this case, we further do not want to make the unwarranted assumption that the subject NP found by the parser properly contains the named-entity found by the named-entity tagger. Therefore, we use the `select-wide::` axis to indicate that the named-entity which will serve as our candidate answer need only overlap with the sentential subject.

How do we map from questions to queries like this? For now, we use hand-crafted patterns, but we are currently working on using machine learning methods to automatically acquire question-query mappings. For the purposes of demonstrating the utility of XIRAF to QA, however, it is immaterial how the mapping happens. What is important to note is that queries utilizing the Stand-Off axes arise naturally in the mapping of questions to queries against corpus data that has several layers of linguistic annotation.

6 Conclusion

We have described a scalable and flexible system for processing documents with multi-dimensional markup. We use stand-off XML annotation to represent markup, which allows us to combine multiple, possibly overlapping annotations in one XML file. XIRAF, our framework for managing the annotations, invokes text processing tools, each accompanied with an input descriptor specifying what data the tool needs as input, and a wrapper that converts the tool's output to stand-off XML. To access the annotations, we use an efficient XPath/XQuery engine extended with new Stand-Off axes that allow references to different annotation layers in one query. We have presented examples of such concurrent extended XPath queries in the context of our corpus-based Question Answering system.

Acknowledgments

This research was supported by the Netherlands Organization for Scientific Research (NWO) under project numbers 017.001.190, 220-80-001, 264-70-050, 612-13-001, 612.000.106, 612.000.207, 612.066.302, 612.069.006, 640.-001.501, and 640.002.501.

References

W. Alink. 2005. XIRAF – an XML information retrieval approach to digital forensics. Master's thesis, University of Twente, Enschede, The Netherlands, October.

P.A. Boncz, T. Grust, S. Manegold, J. Rittinger, and J. Teubner. 2005. Pathfinder: Relational XQuery Over Multi-Gigabyte XML Inputs In Interactive Time. In *Proceedings of the 31st VLDB Conference*, Trondheim, Norway.

P.A. Boncz. 2002. *Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications*. Ph.d. thesis, Universiteit van Amsterdam, Amsterdam, The Netherlands, May.

F.J. Burkowski. 1992. Retrieval Activities in a Database Consisting of Heterogeneous Collections of Structured Text. In *Proceedings of the 1992 SIGIR Conference*, pages 112–125.

A. Dekhtyar and I.E. Iacob. 2005. A framework for management of concurrent xml markup. *Data Knowl. Eng.*, 52(2):185–208.

S. DeRose. 2004. Markup Overlap: A Review and a Horse. In *Extreme Markup Languages 2004*.

T. Grust. 2002. Accelerating XPath Location Steps. In *Proceedings of the 21st ACM SIGMOD International Conference on Management of Data*, pages 109–120.

I.E. Iacob, A. Dekhtyar, and W. Zhao. 2004. XPath Extension for Querying Concurrent XML Markup. Technical report, University of Kentucky, February.

V. Jijkoun, E. Tjong Kim Sang, D. Ahn, K. Müller, and M. de Rijke. 2005. The University of Amsterdam at QA@CLEF 2005. In *Working Notes for the CLEF 2005 Workshop*.

K.C. Litkowksi. 2003. Question answering using XML-tagged documents. In *Proceedings of the Eleventh Text REtrieval Conference (TREC-11)*.

K.C. Litkowksi. 2004. Use of metadata for question answering and novelty tasks. In *Proceedings of the Twelfth Text REtrieval Conference (TREC 2003)*.

P. Ogilvie. 2004. Retrieval Using Structure for Question Answering. In *The First Twente Data Management Workshop (TDM'04)*, pages 15–23.

W. Piez. 2004. Half-steps toward LMNL. In *Proceedings of the fifth Conference on Extreme Markup Languages*.

C.M. Sperberg-McQueen and C. Huitfeldt. 2000. GODDAG: A Data Structure for Overlapping Hierarchies. In *Proc. of DDEP/PODDP 2000*, volume 2023 of *Lecture Notes in Computer Science*, pages 139–160, January.

XMark. 2006. XMark – An XML Benchmark Project. `http://monetdb.cwi.nl/xml/`.