Practical applications of stand-off annotation

Martha Larson,[1] Valentin Jijkoun,[1] Jobst Löffler[2] and Erik Tjong Kim Sang[1]

[1] **{larson, jijkoun, erikt}@science.uva.nl**

Intelligent Systems Lab Amsterdam, University of Amsterdam

Kruislaan 403, 1098 SJ Amsterdam, The Netherlands

[2] **jobst.loeffler@iais.fraunhofer.de**

Fraunhofer Institute for Intelligent Analysis and Information Systems

Schloss Birlinghoven, 53754 Sankt Augustin, Germany

Abstract

An information system that makes use of stand-off annotation stores metadata separately from the data they describe. System architectures separate metadata from data in order to cope with heterogeneous annotations or with multimedia formats. This paper discusses some of the practical aspects of implementing an information system with a stand-off architecture. Two systems that use stand-off annotations are described. The first is a prototype radio archive that provides users with content-based access to archived radio broadcasts. This system uses stand-off annotation to store structural metadata describing the broadcasts, which is used for interactive presentation, as well as speech recognition transcripts, which are used for search. The second system is a question answering system that searches a large text corpus in order to identify spans of text that provide answers to user questions. This system uses stand-off annotation to store metadata generated by a series of different linguistic analysis tools. The final section of the paper treats practical aspects of implementing a retrieval system for a diachronic language corpus. Similarities and differences with the prototype radio archive and the question answering system are discussed.

Ein Informationssystem, das stand-off-Annotation verwendet, speichert Metadaten getrennt von den eigentlichen Daten, die durch die Metadaten beschrieben werden. Systemarchitekturen trennen Metadaten von Daten, um die Handhabung von heterogenen Annotationen oder multimedialen Datenformaten zu ermöglichen. Dieser Beitrag diskutiert einige praktische Aspekte der Implementierung von Informationssystemen mit einer stand-off-Architektur. Zwei Anwendungssysteme, die stand-off-Annotationen einsetzen, werden beschrieben. Das erste ist der Prototype eines Radioarchivs, das dem Benutzer den inhaltsbasierten Zugang zu archivierten Radiosendungen ermöglicht. Das System benutzt stand-off-Annotation einerseits zur Speicherung struktureller Metadaten, die zur interaktiven Darstellung der Radiobeiträge am Benutzerarbeitsplatz eingesetzt werden. Zum anderen wird stand-off-Annotation hier verwendet, um Spracherkennungstransskripte zu verwalten, die vom Benutzer für die inhaltsbasierte Suche im Radioarchiv genutzt werden. Das zweite System ist ein Frage-Antwort-System, das einen großen Textkorpus durchsucht. Das Ziel ist die Identifizierung von Textbereichen, die Antworten auf die vom Benutzer gestellten Fragen geben. Dieses System setzt stand-off-Annotation für die Speicherung von Metadaten ein, die von einer Reihe von verschiedenen linguistischen Analysewerkzeugen erzeugt werden. Der abschließende Abschnitt dieses Beitrags diskutiert praktische Gesichtspunkte der Umsetzung eines Retrievalsystems für einen diachronischen Sprachkorpus. Ähnlichkeiten und Unterschiede der beiden besprochenen Anwendungssysteme, Radioarchiv und Frage-Antwort-System,werden erläutert.

The architecture of many information systems is designed to support a fundamental division between source data and metadata describing the source data. In the case of multimedia systems, this division is necessary because it is not possible to represent annotation of multimedia content in the same format as multimedia essence. In the case of systems handling character data, the separation of data and data description makes it possible to combine many sources of annotation. The combination of multiple varieties of annotation is known as multidimensional markup. Any annotation that is stored separately from the original content can be referred to as "stand-off annotation." Although it is not necessarily the case, stand-off annotation is often XML.

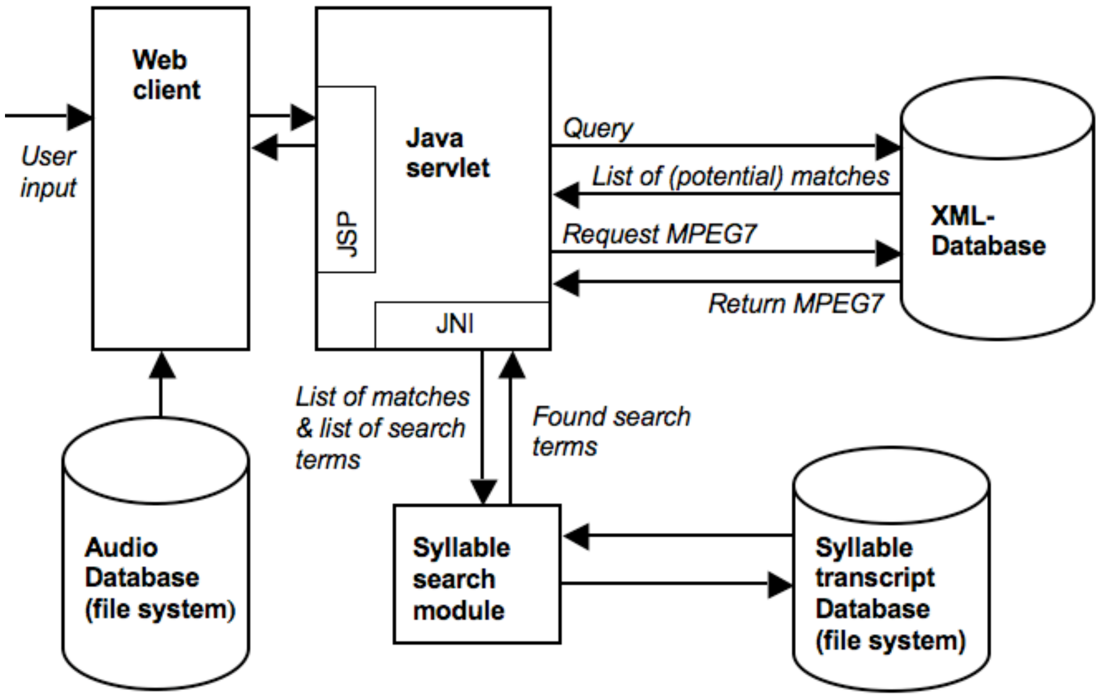
Even though a separation between data and metadata is convenient and often necessary, systems using stand-off annotation face particular challenges when it comes to providing a mechanism for searching their contents. Solutions that maintain the synchronization between data and data-description and the alignment between various sources and levels of description must be provided. This paper discusses some of the practical aspects of addressing the challenges of implementing an information system based on an architecture that makes use of stand-off annotation.

The paper has three parts. In the first two sections, two information systems implemented using stand-off annotations are described, a prototype radio archive and a question answering (QA) system. The systems were designed to provide users with access to two different kinds of information, but both exhibit the principles of architecture and the query mechanisms necessary to put stand-off annotation to practical use. The third section of the paper discusses the issues involved in using stand-off annotation to encode metadata describing a diachronic language corpus. This section focuses, in particular, on some of the practical challenges that can be expected to arise in the design and implementation of an appropriate architecture and the accompanying retrieval mechanism.

1 Prototype radio archive

The prototype radio archive was developed by the Fraunhofer Institute for Intelligent Analysis and Information Systems and was commissioned by Deutsche Welle and Westdeutscher Rundfunk. The system provides content-based access to radio broadcasts, which are stored internally in .mp3 format. These radio broadcasts are annotated with both formal metadata and content metadata. The formal metadata comprise items that can be characterized as bibliographic or production data, including archive ID, title and date of broadcast. The content metadata are transcripts of the spoken content of these radio shows, which are generated by automatic audio processing techniques, including automatic speech recognition. Because speech retrieval makes use of speech recognition transcripts there is significant similarity between the architecture of the prototype radio archive and the architecture of a text retrieval system.

The system architecture of the prototype radio archive is illustrated in Figure 1. The fundamental separation of the audio essence and the metadata that describes it is clearly evident. At the lower left hand corner the audio database containing the audio essence is pictured. At the right side the metadata is stored in two separate systems, some metadata is stored in an XML database in MPEG-7 format and some is stored in a file system. A fundamental principle behind this architecture is that the audio essence and the metadata are associated by means of time codes that reflect the offset with respect to the beginning of the audio file. It is an interesting characteristic of this system that there are two different types of metadata, MPEG-7 metadata and syllable transcript metadata and that these are encoded in two different formats. Again, these metadata are synchronized using time code offsets from the beginning of the audio file. In the following discussion, each sort of metadata is discussed in turn, and then the process used for querying the system is described.



Web client
User input
Java servlet
JSP
JNI
Query
List of (potential) matches
Request MPEG7
Return MPEG7
XML-Database
List of matches & list of search terms
Found search terms
Audio Database (file system)
Syllable search module
Syllable transcript Database (file system)

Figure 1: Architecture of the Prototype Radio Archive

1.1 MPEG-7 Metadata

MPEG-7 is an ISO/IEC standard that was developed by a committee called the Moving Pictures Expert Group to describe multimedia [13][14]. It is a metadata standard that is used in order to describe multimedia content. MPEG-7 was designed to meet the need for a description standard that encodes metadata necessary for a wide range of applications in a way that allows for sufficient flexibility to admit interpretation of meaning.[1] Effectively, a MPEG-7 document consists of metadata that represent multimedia content as a set of building blocks capable of representing multimedia at various levels of abstraction. MPEG-7 was chosen for use in the prototype audio archive in order to ensure interoperability of the metadata in this system with that in other systems. In particular, the prototype makes use of three Description Schemes (DS).[2]

[1] http://www.chiariglione.org/mpeg

[2] It is important to note that the full potential of MPEG-7 is not exploited within this system. In particular, although we are using MPEG-7 to represent spoken content, we do not exploit of Spoken Content Description Tools.

The first DS is CreationInformation DS, which is used to encode the formal metadata. The formal metadata (e.g. title) are created at production time and are supplemented with additional metadata (e.g. broadcast date, archive number) by human archivists when the audio essence is archived.

The second and the third Description Schemes are used to encode automatically generated metadata that were generated by modules of the iFinder SDK media analysis toolkit developed at Fraunhofer IAIS. In the case of the prototype radio archive, the iFinder toolkit has been used to analyze radio broadcasts, but the toolkit also contains capabilities for video analysis and the architecture described here can be supplemented in order to implement a video retrieval system.

The second Description Scheme is the Segment DS, which forms the basis of an abstract type that describes audio segments. The audio segments correspond to a temporal span within a radio broadcast. The segments are generated by an automatic segmentation module, which places segment boundaries at points at which there is a sudden change in the quality of the audio signal. These segment boundaries correspond to speaker changes or to the change from speech to music or from music to speech. A second classifier processes the resulting segments and labels them as containing speech or containing non-speech. This label is stored in the structural description of the segment.

Finally, SemanticBase DS is the third Description Scheme and is used to represent conceptual information about the audio. In the case of the prototype audio archive, this conceptual information pertains to events that occur in the audio. Three sorts of events can be described in this system, a speaker event, a segment of audio spoken by a particular speaker, a jingle event, a segment of audio where a particular radio jingle is played and a keyword event, a segment of audio in which a particular keyword is spoken. Audio events are detected and labeled by classifiers trained on training data drawn from a representative set of radio broadcasts. Speakers are not identified with their names, but rather with an identification number which associates all the segments spoken by the same speaker within the program. The number of speakers in the archive is prohibitively large, too large to make it possible to train classifiers to identify speakers by name within the scope of the prototype radio archive system. The speaker identification number is the output of a speaker clustering algorithm, which groups segments into classes according to acoustic similarity. For jingles, the situation is different than for speakers. Here, classifiers have been trained to identify particular jingles by name.

Division of audio into homogenous segments and the labeling of these segments with identification labels reflecting their content often called diarization. Diarization is commonly undertaken preceding speech recognition, since can serve to prevent transcripts from being generated for non-speech segments. Diarization is an important step in rich transcription, which produces speech recognition transcripts enhanced with additional information about audio content. More information about diarization in the prototype radio archive can be found in [9].

An event is associated with a MediaTimeCode. It is this time information that allows the event, which is a description of the semantic content of the radio broadcast, to be associated with the audio essence, i.e. the radio broadcast itself. Likewise, time information (time offsets) associates a conceptual event with the physical structure of the audio, the audio segment in which the event occurred.

1.2 Syllable transcripts

The syllable database contains the speech recognition transcript generated for each broadcast. The transcript is not a conventional word transcript, but rather consists of syllables. Each syllable has a corresponding time code, which indicates when that syllable was pronounced in the radio broadcast. As in all cases, the offset is measured with respect to the beginning of the radio program. An excerpt of a syllable transcript is represented in

```
f_r_aI_    002150
h_aI_t_    002372
Q_U_n_     002623
g_l_aI_C_  002866
b_e:_      003110
r_E_C_     003378
t_I_       003522
g_U_N_     003778
```

Figure 2: Excerpt of syllable transcript with time offsets

The prototype radio archive uses syllable transcripts generated by a speech recognizer to give users access directly to the spoken content of radio broadcasts. The user can input a keyword and the system will search for the time points at which that keyword was uttered in the spoken audio contained in the archive. Instead of using more conventional word transcripts to represent the spoken content of radio broadcast, the system makes use of syllable transcripts that were generated by a syllable-based speech recognition system. Using syllable transcripts makes it possible for the system to search for words in the speech recognition transcripts that were not originally included in the vocabulary of the speech recognition system. Although the vocabularies of speech recognizers are large, they are finite and in practice often fail to include rare person and place names important for the work of archivists. In order to circumvent the restriction of speech recognizer vocabularies, the spoken word is transcribed as a string of syllables, which can be thought of as the basic unit of pronunciation. With a relatively compact inventory of syllables, in this case 5000, it is possible to reconstruct words and thus identify places in the audio where these words were spoken. As can be seen in Figure 2, the syllables are represented as strings of phonemes. Phonemes are the basic unit of sound of a language and they combine to form syllables. When language is transcribed in terms of its basic units of sound it is possible to perform search based on phonetic similarity between the search word and the speech recognition transcript. Basically, the system scans the syllable transcripts for places in which they indicate that the audio sounds like the query word. In the prototype radio archive, keywords are found in the syllable transcripts by making a “sounds alike” comparison. This comparison scans through the syllable transcripts sequentially and determines at which time codes a syllable string begins that has sufficient phonetic similarity to the query word.

The sequential scan is accomplished by moving a window through the text and checking at each syllable position if the syllables in the window match the query word closer than a given match threshold. The system has to check for every syllable if that syllable could be the beginning of a query word since the syllable-based recognizer outputs a syllable string only and does not detect word boundaries. Although certain techniques can be applied to approximate the sequential scan thus allowing the syllable transcripts to be represented in a conventional index, the prototype radio archive was interested in understanding the performance ceiling of an (near) exhaustive search. For this reason, no indexing techniques, which might have been used to accelerate the search process, were implemented.

1.3 Querying the prototype radio archive

The challenge in designing the mechanism used to query the prototype radio archive lay in coordinating the search in the MPEG-7 database with the search in the syllable transcripts. In the query mask the user can enter words or phrases that must be contained in the title of the program or a range of broadcast dates or archive numbers, thus restricting the results list using the formal metadata. The user can also enter a keyword that should be found in the spoken audio of the archive content. It would be possible to issue two queries, one to the XML database requesting programs that conform to the restrictions on the formal database and one to the syllable database requesting programs that contain the spoken keyword. The results of the two queries would then be intersected to build the hit list presented to the user. This manner of implementation would not be optimal with respect to speed, however, since the sequential scan of the syllable transcripts makes searching the syllable database markedly more time consuming than searching the XML database.

For this reason, the query is decomposed into a part concerning formal metadata and a part concerning content metadata and is treated in multiple steps, which are illustrated by the arrows in Figure 1. First the part of the query concerning the formal metadata is sent to the XML-database and a list of the archive numbers of programs with formal-metadata matches is returned to the system, as illustrated by the arrows labeled “Query” and “List of (potential) matches.” Then this list of archive numbers is sent along with the part of the query concerning the content metadata to the syllable search modules, as illustrated by the arrow labeled “List of matches & list of search terms.” The syllable search module accesses the syllable transcript database and sequentially searches the program transcripts corresponding to the numbers in the list. It then returns to the system a subset of the list of archive numbers and time codes within the corresponding syllable transcripts at which the spoken keyword was found (cf. arrow labeled “Found search terms”). In the final step, the XML-database and the audio database are accessed to retrieve the MPEG-7 description and the audio essence for display to the user.

2 Question Answering System

The Information and Language Processing Group at ISLA at the University of Amsterdam is developing an automatic system for Question Answering (QA). In a typical setup (such as at the TREC[3] benchmarking conference), a QA-system has to find a precise answer (or answers) to a natural language question in a large fixed collection of text documents, usually in archives of newspaper and newswire articles.

A traditional approach to the textual QA task [17] builds on a three-step pipeline. First, from a natural language question, the QA-system automatically generates a retrieval query, which is used to *retrieve* a list of documents (or document passages) that are likely to contain answers

[3] See http://trec.nist.gov/data/qa.html

to the question. Second, the documents or passages identified at the retrieval step are further analyzed and candidate answers are *extracted*. Finally, the QA-system *combines* answers extracted at the previous step, ranks them and returns the best one(s) as a response to the question.

The second step, answer extraction, typically involves syntactic and semantic analysis of text passages, at various depths and involving various text annotation tools and linguistic resources. The most commonly used tools include part-of-speech taggers, named entity taggers, sentence and clause splitters, phrase chunkers, syntactic dependency and phrase parsers and shallow semantic parsers. Two approaches to using such annotation tools can be identified: (1) running the tools "on the fly" for all text passages identified at the retrieval stage, and (2) running the tools for the text of the entire document collection off-line, before the questions are known. The second approach allows the system to avoid the time- and resource-expensive text annotation step at runtime by moving it offline; ILPS's QA-system uses this offline approach.

Flexible and efficient access to the annotations of the text during the answer extraction step is crucial for a QA-system. A straightforward approach is to store annotations using XML and to access the data use XPath/XQuery,[4] standard query languages for accessing XML data. However, this approach is problematic. Whereas annotations of individual tools (taggers, parsers) can easily be stored in XML, independent tools often produce incompatible annotations due to different text tokenization, different treatment of encodings or analysis errors. On the other hand, a QA-system often requires simultaneous access to several annotations. For example, when answering a question "Who shot JFK?" the system will invoke a query which will look for passages containing the words "shot" and "JFK," and then will try to identify human subject present in the sentences of the retrieved passages. The system might use additional linguistic clues, e.g., specifying that "shot" should be a verb and not a noun, or using named entity information to indicate that "JFK" should refer to a person and not to the airport, or enforcing that the extracted answer candidate should be a person name occurring as the subject of a verb whose direct object is "JFK."

2.1 Storing text annotations

We solve the problem of unifying the access to the output of different text analysis tools by adopting *stand-off annotation*. We consider each tool's output as a separate *annotation layer* of the same textual data, *the blob* (Binary Large OBject in the database parlance). Individual annotation layers use conventional XML, where every element is supplied with two additional attributes: the start and end offsets of the part of the blob to which this element refers. For example, in Figure 3 and Figure 4 we show the simplified named entity and syntactic phrase tree annotation layers of the sentence "The evidence supporting the claim that LHO killed JFK does seem pretty conclusive."

[4] See http://www.w3.org/TR/xpath and http://www.w3.org/TR/xquery/

```
The evidence supporting the claim that <NE type="PERSON" start="39"
end="41">LHO</NE> killed <NE  type="PERSON" start="50"
end="52">JFK</NE> does seem pretty conclusive .
```

Figure 3: Named Entity annotation layer. In the QA-corpus each name has been marked with a tag that contains attributes specifying its type and its byte offsets

```
…
<phrase type="NP" start="24" end="">
  <word tag="DT" start="24" end="26">the</word>
  <word tag="NN" start="28" end="32">claim</word>
  <phrase type="SBAR" start="34" end="80">
    <word tag="IN" start="34" end="37">that</word>
    <phrase type="S" start="39" end="80">
      <phrase type="NP" start="39" end="41">
         <word tag="NNP" start="39" end="41">LHO</word>
      </phrase>
      <phrase type="VP" start="43" end="80">
        <word tag="VBD" start="43" end="48">killed</word>
        <phrase type="NP" start="50" end="52">
          <word tag="NNP" start="50" end="52">JFK</word>
        </phrase>
      </phrase>
    </phrase>
  </phrase>
</phrase>
…
```

Figure 4: Syntactic annotation layer. In the QA-corpus phrases and words have been marked with tags that contain attributes specifying their types and their byte offsets.

Note that the actual text data in different annotation layers can be different: text analysis tools apply different rules regarding tokenization, punctuation, whitespace, encodings, accents, etc. Therefore we perform an additional *alignment* step: the text data in the XML output of each annotation tool is compared to the original blob taken from the source document and a simple algorithm based on string edit distance computes "true" offsets of annotation elements in the original blob. E.g., in the example above the *start* and *end* attributes refer to offsets in the original blob, not in the text content of the tool's output.

2.2 Accessing the annotations

Although we store annotations in different XML trees, we would still prefer to access them with a single query rather than issuing many queries. If one query were issued for each XML tree, then the results would need to be combined, necessitating comparison of the offsets. At this point, we employ two methods for accessing our data: using the XQuery extension of MonetDB [15] and using our in-house extension of the XQuery Nux[5] library for Java.

[5] See http://dsd.lbl.gov/nux

2.2.1 MonetDB

MonetDB [15] is a relational database management system that supports XML databases and provides efficient XQuery access to XML documents using standard relational algebra operators on element offsets. XML structure of documents is stored in relational databases in a way that allows for a very efficient implementation of XQuery features. Moreover, MonetDB currently supports multidimensional markup [2] with stand-off annotation and provides access to multi-layered documents via additional axes in XQuery.

Specifically, MonetDB/XQuery implements *select-narrow, select-wide, reject-narrow* and *reject-wide* XML axes (inspired by [3]), in addition to the standard *child, parent, ancestor, descendant,* etc. axes. Essentially, the new axes provide a direct access to annotation elements that overlap with (or contain) any current element, even if they belong to a different annotation layer. Thus, these axes allow a user to make a connection between different sorts of annotations and to deal with overlap in a transparent way.

To give a simple example, the query `//C/select-wide::*` returns a list of all XML elements that overlap (i.e., have at least one character in common) with an element `C`, whether they belong to the same annotation layer as `C` or not. To give a more realistic example, let's consider sentence annotations from Section 2.1 above. The following MonetDB query finds person names that occur as parts of syntactic objects of verbs:

```
//phrase[@type="VP"]/phrase[@type="NP"]/select-wide::NE[@type="PERSON"]
```

This query first locates all verb phrases ("VP") which have noun phrases ("NP") as children, and looks for person names that occur "near" these noun phrases. Note that in this case the overlap axis ("select-wide") allows us to retrieve elements even in presence of annotation errors (e.g., named entities crossing phrase boundaries).

2.2.2 Extension to Java XQuery library

As an alternative to MonetDB/XQuery with multidimensional markup, we developed an extension to Nux' the standard Java library for XQuery processing. When accessing an XML document with multiple annotations, we load separate XML files containing layers of stand-off annotation of the same data, and join them in a single new XML object (DOM tree) as children of a new root element. The new document can be accessed with the standard XQuery facilities. Additionally, in order to allow users to transparently jump between annotation layers in the document, we use the open architecture of Nux and implement an XQuery function *stand-off:wide* that retrieves XML elements overlapping with the current element, irrespective of the annotation layer, based on offsets specified with *start* and *end* attributes. For example, the following XQuery is equivalent to the MonetDB query above:

```
stand-off:wide(//phrase[@type="VP"]/phrase[@type="NP"])/NE[@type="PERSON"]
```

2.3 Experiences with Storage and Retrieval

We have been using XML retrieval with stand-off annotations for more than two years now. Although our experiences are positive, we have had to handle some unforeseen problems. First, we had to deal with the fact that our annotation tools sometimes modified the processed text. Punctuation marks were separated from words (by the sentence splitter) or changed to different tokens to avoid interpretation problems (parser). Our annotation system needed to realign the output of the tools with the text blobs in order to fit the annotations in the XML file.

One of the design decisions was to store annotations system internally using UTF-8. We also decided to use byte offsets instead of character offsets because in this way access to blobs of UTF-8 data in the memory can be implemented significantly more efficiently. We discovered that alignment was problematic since some annotation tools converted the text encoding from UTF to Latin 1, thus causing a mismatch between the byte offset encodings [6]. We solved this problem by writing wrappers for the tools that guaranteed format consistency.

An even more serious problem is scaling. As discussed above, we store the text and annotations in MonetDB, an XML database engine with full XQuery support and multidimensional markup. Our recent work [7] involves text corpora of dozens of gigabytes. While we have been able to store these corpora using MonetDB, we have up until now failed to find a stable method for retrieving data from the corpus. It seems that the problems are due to the sheer size of our data: MonetDB has not been tested with collections of that size before. Our alternative method for accessing stand-off annotations (Section 2.2.2) makes it possible to circumvent the scalability problem using the fact that the answer extraction step of our QA-system works with the collection on the level of newspaper articles, typically several kilobytes of text. For such small units, merging annotation layers in memory every time an article is accessed is computationally affordable.

3 Information system for a diachronic language corpus: Implementation issues

In this section, we summarize the challenges that a diachronic language corpus poses for a system designed with an architecture that accommodates stand-off annotation and present a view on how to approach these challenges when implementing a retrieval system. A diachronic language information system contains historical texts in digital form that have been enriched with a variety of annotations. Such a system supports researchers by allowing them to perform cross-time/cross-text comparisons. The system makes it possible to find answers to complex queries concerning both qualitative and quantitative aspects of the diachronic language corpus it contains. Digital diachronic language corpora are an indispensable resource for language researchers, including historical linguists, lexographers and researchers engaged in the study of literature or social science. A description of the kind of queries that can be answered by such a corpus can be found in [11], which outlines a plan for *Deutsch Diachron Digital*, a diachronic corpus of German spanning texts from Old High German to modern German. The kinds of queries that will aid linguists in their research include queries concerning the extent to which ligatures are used across the boundaries of compound components, queries concerning the change in the spelling of a word over time and queries concerning the variation of verb position across time and across dialects. These simple examples are intended to give an impression of the utility of a diachronic language information system for supporting linguistic research. In practice, queries used by researchers can be significantly more specific and complex than these examples.

The challenges in the design and implementation of an information system for *Deutsch Diachron Digital* are typical of those confronting a diachronic language corpus. These challenges are described in detail in [12][11] and outlined here. The diachronic texts are described with different layers of annotation including sentence structure, page layout, syntactic structure, categorical information and translation. A typical annotation layer for a diachronic corpus is a normalized layer in which the original text is standardized, including abstracting away from dialectual variation and writing out abbreviations. Another typical characteristic of a diachronic language corpus is that it contains multiple versions of the same text that must be associated with each other on a fine-grained scale. These versions include manuscript copies, different editions as well as translations. Alignment must be maintained between the different levels of information and it must be possible to search the corpus for text segments and annotations that stand in particular relationships to each other. The annotations generally form tree structures, but these tree structures often partially overlap and provide difficulty for conventional XML representation. Practically, the corpus must be extendible with new texts and new annotations. It also must be able to include multimedia documents, in particular scans of the original manuscripts.

3.1 Lessons learned from the prototype radio archive relevant to a diachronic language information system

A diachronic language corpus will contain text material, but, as just mentioned, it will also most likely contain multimedia material, i.e. scanned images of original manuscripts. For this reason, a retrieval system for a diachronic language corpus is in effect forced to respect the division between essence and metadata as the prototype radio archive does. Separating data from metadata carries with it the additional advantage that it is possible to more easily extend annotation layers and add additional annotation layers. In the prototype radio archive, time-offsets synchronize the syllable transcripts and the MPEG-7 metadata with the audio essence. In the diachronic language corpus, two sorts of offsets are necessary. First, those offsets that provide synchronization of metadata with images of manuscripts, and second, those offsets that synchronize character metadata with annotation layers.

A commonality between the prototype radio archive and the diachronic language corpus is that both have document-level metadata containing bibliographic information. If the query specifies constraints on document-level metadata, these constraints can be used to restrict the set of documents over which the more time-consuming content-based search is performed. It should be noted that if a system is designed in such a way that time-intensive searches occur only on a subset of the entire collection, the system will scale more readily to larger collections.

3.2 Lessons learned from the question answering system relevant to a diachronic language information system

A diachronic language information system will contain multiple layers of often partially overlapping annotation and in this way is similar to the question answering system. Like the question answering system, a diachronic language information system can exploit the possibility of physically storing annotation layers in different files and merging the annotations at query time. A diachronic language information system must also make the design decision of what sort of a system-internal offsets to apply. Although character offsets have intuitive appeal, byte offsets will simplify memory access for data with multi-byte character encodings, e.g. UTF.

Many of the storing and querying facilities necessary for a diachronic language information system could be implemented with MonetDB, making use of XQuery capabilities extended with operators used to access multi-dimensional markup. We would like to note that at the time of writing, if fast retrieval is necessary, it is necessary to use a separate retrieval engine for text search functionality rather than the database itself. However, MonetDB is under continuous development and new functionalities are being integrated on an on-going basis.

3.3 Additional issues

An information system for a diachronic language corpus will be inherently complex due to the heterogeneity of the data and of the annotations it contains and to the needs of the language researchers that will use it. There are several important issues that arise for a diachronic language system that are not relevant for either the prototype radio archive or for the question answering system. First, in order to provide widely available service, a diachronic language information system must be able to support multiple simultaneous users; as research systems, neither the prototype radio archive nor the question answering systems integrate such functionality. Second, a primary sort of user query will concern quantitative properties of the corpus; thus, a diachronic language system must produce corpus statistics. Third, a diachronic language information system will need to accommodate references between different documents it contains, for example, aligning two editions of the same text. Neither the radio prototype corpus nor the QA-corpus contains information in their metadata layers about links between documents within the corpus. The query mechanisms for both systems exploit the fact that time-consuming search steps (syllable transcript scan in the radio archive prototype and answer extraction in the QA-system) are restricted to subsets of the entire corpus. If a corpus contains references between documents, it will not be possible in a straightforward manner to select a subsection of the corpus for separate search. MonetDB would, at the time of writing, be able to handle and entire text corpus, were its size not to exceed 1G. For larger corpora, other solutions must be implemented.

4 Conclusion

Information systems using stand-off annotation need appropriate architectures and appropriate mechanisms for querying. This paper discussed two systems that implement stand-off architectures. These systems served as examples to illustrate specific design decisions entailed by the use of stand-off annotations and the practical consequences of these decisions. A prototype radio archive was discussed in which stand-off annotations needed to be stored in two different formats. Time offsets were used to synchronize the annotations with each other and with the audio essence. A question answering system was discussed in which a text corpus was annotated with multiple layers of metadata. Byte offsets were used for synchronization and a special query mechanism was implemented to deal with the problem of overlap in the different annotation layers. The paper concluded with a discussion of issues that arise when implementing an information system for a diachronic language corpus.

5 References

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

[2] Alink, W, Jijkoun, V., Ahn, D., de Rijke, M., Boncz, P. and de Vries, A. “Representing and Querying Multi-dimensional Markup for Question Answering.” In *Proceedings of the 5th Workshop on NLP and XML (NLPXML-2006) Multi-Dimensional Markup in Natural Language Processing*, 2006.

[3] Burkowski, F.J. “Retrieval Activities in a Database Consisting of Heterogeneous Collections of Structured Text.” In *Proceedings of the 1992 SIGIR Conference*, pages 112-125, 1992.

[4] Chang, S.-F. , Sikora, T. and Puri, A. “Overview of the MPEG-7 standard.” *IEEE Transactions of circuits and systems for video technology*, vol. 11, no. 6, pp. 688-695, 2001.

[5] Dipper, S., Faulstich, L., Leser, U. and Lüdeling, A. “Challenges in Modelling a Richly Annotated Diachronic Corpus of German.” In *Proceedings of the workshop on XML-based richly annotated corpora*, 2004.

[6] Jijkoun, V., van Rantwijk, J., Ahn, D., Tjong Kim Sang, E. and de Rijke, M. “The University of Amsterdam at QA@CLEF 2006.” In *Working Notes for the CLEF 2006 workshop*, 2006.

[7] Jijkoun, V., Hofmann, K., Khalid, M.A., van Rantwijk, J. and Tjong Kim Sang, E. “The University of Amsterdam at QA@TREC 2007.” To appear in the *Proceedings of the Sixteenth Text Retrieval Conference (TREC 2007)*, NIST, 2007.

[8] Larson, M., Eickeler, S. and Köhler, J. “Supporting Radio Archive Workflows with Vocabulary Independent Spoken Keyword Search.” In *Proceedings of the SIGIR 2007 Workshop Searching Spontaneous Conversational Speech*, 2007.

[9] Larson, M. and Köhler, J. “Structured Audio Player: Supporting Radio Archive Workflows with Automatically Generated Structure Metadata.” In *Proceedings of RIAO 2007.*

[10] Löffler, J., Biatov, K., Köhler, J. “Automatic extraction of MPEG-7 audio metadata using the media asset management system iFinder.” In *Proceedings of the AES 25th International Conference: Metadata for Audio*, 2004.

[11] Lüdeling, A., Poschenrieder, T. and Faulstich, L.C. “DeutschDiachronDigital – Ein diachrones Korpus des Deutchen.” In *Jahrbuch für Computerphilologie 2004*, pages 119-136, 2005.

[12] Lukas C. Faulstich, L.C., Leser U. and Lüdeling, A. “Storing and Querying Historical Texts in a Relational Database.” *Informatik-Bericht Nr.176 des Instituts für Informatik der Humboldt-Universität zu Berlin*, February 2005.

[13] Manjunath, B.S., Salembier, P.P. and Sikora, T. *Introduction to MPEG-7: Multimedia Content Description Interface*, John Wiley and Sons, 2002.

[14] Martinez, J.M., Koenen, R., and Pereira, F. “MPEG-7: the generic multimedia content description standard, part 1.” *IEEE Multimedia*, vol. 9, no. 2, pages 78-87, 2002.

[15] MonetDB. Website: http://www.monetdb.nl/, 2007.

[17] Monz, C. *From Document Retrieval to Question Answering*. PhD thesis. University of Amsterdam, 2003.

6 Acknowledgements

We would like to acknowledge Deutsche Welle and WDR, who commissioned the prototype radio archive. Martha Larson is supported by the E.U. IST programme of the 6th FP for RTD under project MultiMatch contract IST-033104. Valentin Jijkoun is supported by the Netherlands Organization for Scientific Research (NWO) under project number 640.001.501. Erik Tjong Kim Sang is supported via grants supplied via the NWO project IMIX (264-70-050) and the STEVIN project Cornetto.