Focused Search in Digital Archives

Junte Zhang[1] and Jaap Kamps[1,2]

[1] Archives and Information Studies, Faculty of Humanities, University of Amsterdam
[2] ISLA, Faculty of Science, University of Amsterdam

Abstract. We present a system description for an archival information system with three different approaches to gain online access to digital archives created in the metadata standard Encoded Archival Description (EAD). We show that an aggregation-based system can be developed on archival data using XML Information Retrieval (XML IR). We describe the different stages and components, such as the indexing of the digital finding aids in an XML database, the subsequent querying and retrieval of information from that database, and the eventual delivery of that information to the users in a contextual interface.

1 Introduction

Cultural heritage (CH) information from libraries, museums and archives can be increasingly found online. In the past, the physical CH artifacts, like books, paintings or a personal letter, were described and catalogued in paper finding aids by curators. For example, a user who was looking for a personal letter in a collection created by a historical person, had to go to an archive to find that letter by consulting paper finding aids and the archivist. Nowadays, with the advent of digital finding aids to provide online access to these (unique) physical artifacts, that is no longer needed. A major benefit for users is that CH materials are disclosed more effectively and efficiently both in terms of time and effort.

Several metadata schemas are used to create the digital finding aids, such as Dublin Core, MARC, and increasingly, the international standard Encoded Archival Description (EAD). The archives, but also manuscript libraries and museums, are expanding their digital resources by adopting EAD in XML and putting them online as digital archives, which means that structural CH information can be exploited on the Web for web services. The state-of-the-art online archival finding aids in EAD are a nearly one-to-one mapping of paper finding aids. A distinct property of the old paper finding aids and hence also the new ones in EAD, is that these files are long in content and complex in structure with very deep nesting of the elements in the XML tree hierarchy.

This paper outlines a system description for README[1]—an online archival information system that is able to retrieve information within the archives using three approaches that exploit the granularity and structure of archival finding aids in EAD using XML Retrieval in order to provide focused access (see Section 2). Section 3 continues by explaining the different components of the system

[1] Acronym for "Retrieving Encoded Archival Descriptions More Effectively."

G. Vossen, D.D.E. Long, and J.X. Yu (Eds.): WISE 2009, LNCS 5802, pp. 463–471, 2009.

more in detail. We point to the evaluations of the three system approaches in Section 4, and conclude the paper in Section 5.

2 Related Work

2.1 Encoded Archival Description

An increasing number of archives and manuscript libraries, and also museums, use the international standard Encoded Archival Description (EAD) to encode data that describe unique primary resources in the form of archival materials, such as corporate records and personal (hand-written) papers [5]. These collections may have millions of unique items, which can be in any form or medium.[2]

The archives are organized hierarchically. EAD consists of a set of descriptive elements to describe the archives. The three highest level elements are <EADHEADER>, the optional <FRONTMATTER>, and the archival descriptions in <ARCHDESC>. The components <C$_n$> of the whole are nested in <ARCHDESC>, where $n \in \{01, ..., 12\}$, see Fig. 4. For example, <C02> is the sub-component of <C01>, and so on. A component can also be unnumbered. The EAD files can be deeply nested and lengthy in content with thousands of pages (or more) [5].

There is no shortage of metadata in archival finding aids [4], but is "just a matter of finding the right hook to make them more accessible." XML Information Retrieval techniques can be employed to deal with this problem and be used to maximally and most effectively exploit these 'hooks.' Using this markup, we can zoom into any of them—at the same time index and retrieve them.

2.2 XML Information Retrieval

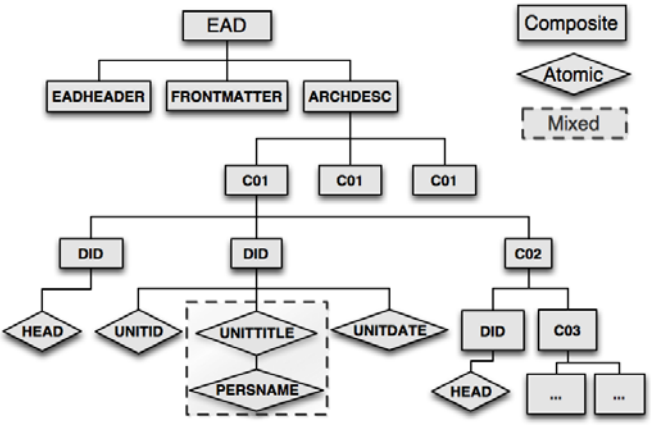
The indexing and retrieving of these 'hooks' (elements) is done using XML Information Retrieval (XML IR), which is a branch of Information Retrieval that deals with the retrieval of arbitrary parts of XML files given the XML structure, and attempts to use the XML markup of documents to the fullest for 'focused' information access by not only providing direct access to a whole document, but also to a part of the document. The structure is exploited to expose information.

As illustrated in [8], structured text retrieval supports the representation and retrieval of the individual document components defined by the logical structure as represented in a hierarchical document, such as an EAD file. This structure can be distinguished in two types of units [8]: (a) atomic units (or 'text content elements') that only contain text and no XML elements, and (b) composite units (or 'nested elements') that contain other units and can be further 'decomposed'. The same is true for EAD, see Fig. 4, where atomic units such as <UNITID>, <UNITTITLE> or <UNITDATE> are represented as leafs and composite units like <DID> are non-leaf nodes. However, we extend this representation with *mixed content* nodes, i.e. elements that contain both text and other elements. An instance of a mixed node could be the composite unit <UNITTITLE> that may have been annotated with a semantic tag like <PERSNAME> (which is allowed in EAD).

[2] For example, plans, drawings, charts, maps, photographs, audio, and video [5].

```
<EAD>
  <EADHEADER>
    [..]
  </EADHEADER>
  <ARCHDESC>
     [..]
    <C01>
      [..]
      <C02>
        [..]
        <C03>
          <DID>
            <UNITID>
               [..]
            </UNITID>
            <UNITTITLE>
               [..]
            </UNITTITLE>
            <UNITDATE>
              [..]
            </UNITDATE>
          </DID>
          [..]
        </C03>
      </C02>
    </C01>
  </ARCHDESC>
</EAD>
```

(a) EAD in XML



(b) EAD as mono-hierarchical schema

Fig. 1. Representation of Encoded Archival Description as mono-hierarchical schema using XML elements as nodes for Information Retrieval

3 README System Description

3.1 Digital Finding Aids in XML

We obtained in total 8,159 finding aids from 3 sources: the National Archives of the Netherlands (NA), the International Institute of Social History (IISH), and the Archives Hub (AH). The statistics are shown in Table 1, which shows the number of files from each source, the distribution of the length of content in bytes (without XML markup) and of the structure in terms of XML tags.

We show that there is strong positive and significant correlation on a 95% confidence interval between content length and XML markup (Spearman's ρ and Kendall's τ, $p < 0.01$, 2-tailed). The correlation in the NA data is very strong, likely due to the length which results in more tags. This correlation is less strong for the finding aids from the AH or IISH, because their finding aids were shorter, and sometimes copy-pasted from legacy data, where the conversion to EAD has not been complete, and hence large chunks of text without XML markup occurs.

3.2 Indexing and Storage

Before the indexing and storage, we preprocess the files to make them strictly well-formed and valid XML—which was a prequisite for indexing in an XML database. Many of the files were not well-formed XML (missing closing tags, wrong nesting), and the ones from the Archives Hub were in SGML. In order to map them to well-formed XML, we bootstrap the files using the SGML to XML

Table 1. Statistics of the archival finding aids, where ** is significant at 0.01 level (2-tailed) using Spearman's rho and Kendall's tau

Source	N	Content (bytes)		Structure (count)		Correlation	
		Mean	Median	Mean	Median	Spearman's ρ	Kendall's τ
NA	2,174	53,571.65	12,974	2,891.46	481	0.9596**	0.8280**
IISH	2,866	11,187.19	1,736	481.93	57	0.7678**	0.5916**
AH	3,119	3,886.96	2,054	117.94	65	0.6958**	0.5310**

converter OSX in OpenJade[3], then process them again in XML Lint[4], and then cleaning them up (like making all tags uppercase) in HTML Tidy[5]. Since we deal with mostly Dutch language data, but for example also French and German, we used the ISO/IEC 8859-1 character encoding.

The system is based on MonetDB with the XQuery front-end Pathfinder [1] and the information retrieval module PF/Tijah [3]. All of our 8,159 finding aids in EAD are indexed into a single main memory XML database that completely preserves the XML structure and allows powerful XQuery querying. We indexed the collection without stopword removal, and used the Dutch snowball stemmer.

3.3 Retrieval Model

For the retrieval of individual and any arbitrary elements, we employ statistical language models (LM) [6], i.e. the probability distribution of all possible term sequences is estimated by applying statistical estimation techniques. The probability of each individual term is calculated using the *maximum likelihood estimate* (*mle*), which corresponds to the relative frequency of a term t_i in an element e, $P_{mle}(t_i|e) = \frac{\text{tf}_{i,e}}{\sum_t \text{tf}_{t,e}}$ where $\text{tf}_{i,e}$ is the term frequency t_i normalized by the sum of all frequencies in an element e.

We estimate the probability that the element model can generate the given query q. By applying Bayes' theorem, this can be obtained by

$$P(e|q) = \frac{P(q|e) \cdot P(e)}{P(q)} \propto P(q|e) \cdot P(e) \tag{1}$$

where $P(q)$ can be ignored for ranking, and the prior $P(e)$ is assumed to be uniform. The query likelyhood (or conditional probability) is based on a model that represents an element using a multinomial probability distribution over a vocabulary of terms. For each element, a model on an element is inferred, such that the probability of a term given that model is $p(t|e)$. The model is then used to predict the likelihood that an element could match a particular query q. We make the assumption that each query term can be assumed to be

[3] http://openjade.sourceforge.net/
[4] http://www.xmlsoft.org/xmllint.html
[5] http://tidy.sourceforge.net/

```
XQuery snippet
let $options := <TijahOptions ir-model="LMS" collection-lambda="0.15" returnNumber="M" />
let $query_text := tijah:tokenize("query terms")
let $query_nexi := concat("//EAD[about(., ", $query_text, ")]")
let $qid := tijah:queryall-id($query_nexi, $options)
let $nodes := tijah:nodes($qid)
```

Fig. 2. XQuery code that illustrates the initialization of system parameters and the use of NEXI for querying. Here, we search in root nodes only, which corresponds to the full text of the document.

```
XQuery snippet
let $result := for $node at $relevance in $nodes
return
<result>
   <rel>{ $relevance }</rel>
   <num>{ (count($node/preceding::*) + 1) }</num>
   <file>{ data($node/ancestor-or-self::EAD/@FILE) }</file>
   [more xpath selections...]
</result>
let $total := count($result)
return <results total="{$total}"> {
   for $res in distinct-values($result/file)
      let $cs-group := $result[file = $res]
      for $cs-group2 at $rank in $cs-group
      where $rank <= N
      order by string($cs-group2/file), number($cs-group2/num), number($cs-group2/rel)
      return
      <out id="{$res}">{ $cs-group2 }</out>
} </results>
```

Fig. 3. XQuery code that illustrates the retrieval of elements according to relevance, grouping of results by file name, and subsequent re-ordering of the retrieved results given the original document hierarchy

sampled identically and independently from the element model. Applying this assumption, the query likelyhood is obtained by multiplying the likelihoods of the individual terms contained in the query:

$$P(q|e) = \prod_{t \in q} P(t|e)^{n(t,q)} \tag{2}$$

where $n(t, q)$ is the number of times term t is present in query q.

To deal with zero probabilities because of non-existing terms in case there is sparse data, smoothing techniques are applied. The retrieval model uses Jelinek-Mercer smoothing, which is a mixture model between the element model and the collection as background model, so

$$P(t|e) = (1 - \lambda) \cdot P_{mle}(t|e) + \lambda \cdot P_{mle}(t|C) \tag{3}$$

where $P_{mle}(t|C) = \frac{ef_t}{\sum_t ef_t}$, ef_t is the element frequency of query term t in the collection C, and the λ is set to 0.15.

3.4 Querying and User Interfaces

We discuss now the three approaches deployed in the README system, which is written in Perl using XHTML, CSS, and JavaScript. The connection with the

(a) Document retrieval

(b) Element retrieval

(c) Aggregation-based retrieval

Fig. 4. An overview of the three approaches in the README system with the query "koude oorlog spionage" (in English: cold war spying)

database server is made in Perl using a socket and XML RPC. We can search between different sources and within a source—the provenance is made clear by showing an icon in front of a result that corresponds to a source. For each retrieval approach, we also present a user interface (see Fig. 4).

Approach 1: Document Ranking. The XML database is queried using XQuery extended with Narrowed Extended XPath I (NEXI) [7]. For document ranking, we provide the root element (the whole document) as target element. The following piece of XQuery code in Fig. 2 illustrates the procedure in PF/Tijah for document ranking that retrieves M number of documents stored in `$nodes`. The corresponding interface is depicted in Fig. 4(a).

Fig. 5. Deeplinking to the result display with dynamic Table of Contents

Approach 2: Element Relevance Ranking. For element relevance ranking (see Fig. 4(b)), we do not provide a structural hint in the form of a target element, hence any EAD element can be retrieved, including the absolute XPath of an element, such as `/EAD[1]/ARCHDESC[1]/DSC[2]/C01[4]/C02[8]/DID[1]`. It describes the position of an element in the XML tree hierarchy. The rest of the procedure is the same as the document ranking as described above.

Approach 3: Aggregation-based Ranking. The approach goes a step further than the standard element relevance ranking as Fig. 3 and Fig. 4(c) show. It takes relevance `<rel>` into account. Any and arbitrary elements can be retrieved. The retrieved elements are returned in original order as in the XML file, by computing the distance of the retrieved element to the root node in `<num>`. We group the retrieved elements by its creator `<file>`. Eventually, all retrieved elements are ordered by these variables with the top N number of elements per archive. In our system we set this to 8, but it can be made dynamic by allowing users to move beyond that threshold. As explained in [9], the aggregation-based approach optimally utilizes the context of the archives.

3.5 Result Delivery

The hitlist is connected to the result display with HTTP parameters using CGI: the query, XPath, source, and file name are always stored in the URL for persistency and to facilitate the analysis of the search logs. The system can deep-link (with the element and aggregation approaches) by rendering HTML anchors for each element using its (unique) XPath as anchor identifier. We deliver a result by physically linking a result to its file, and render its result display with the Table of Contents (ToC) using the SAXON XSLT processor[6]—this is faster than retrieving everything again from the index. There is minimal transformation from the original XML file, because EAD is as much document-centric (directly view-able by users in a browser) as it is data-centric. We use the Yahoo! User

[6] `http://saxon.sourceforge.net/`

Interface Library (YUI)[7] to make the ToC dynamic and enable enhanced interaction, see Fig. 5. The ToC can be dragged and collapsed—making it an extra non-obtrusive tool to locate information within the retrieved file.

4 Evaluation

On the one hand, the system has been (preliminary) evaluated with 9 users, and more details on this study can be found in [2]. On the other hand, we have evaluated the system from a system-focused point of view [10]. The user study showed that the element ranking approach was least appreciated out of the 3. The aggregation-based approach was appreciated the most. However, the retrieval experiment showed that the element ranking approach has far better retrieval performance than the aggregation-based approach, though the aggregation-based approach seems to find more relevant results in the beginning due to the organization of the archives—showing support for the aggregation-based approach.

5 Conclusion

We have formally introduced and described the Retrieving Encoded Archival Descriptions More Effectively (README) system that provides enhanced access to cultural heritage information. The system employs the XML IR method as an alternative, more focused means to gain access to online digital archives, effectively exploiting the structure to search and find valuable information.

Acknowledgments. This research is supported by the Netherlands Organisation for Scientific Research (NWO) under project #639.072.601.

References

[1] Boncz, P.A., Grust, T., van Keulen, M., Manegold, S., Rittinger, J., Teubner, J.: MonetDB/XQuery: A Fast XQuery Processor Powered by a Relational Engine. In: SIGMOD 2006, pp. 479–490. ACM, New York (2006)
[2] Fachry, K.N., Kamps, J., Zhang, J.: Access to archival material in context. In: IIiX 2008, pp. 102–109. ACM, New York (2008)
[3] Hiemstra, D., Rode, H., van Os, R., Flokstra, J.: PF/Tijah: text search in an XML database system. In: OSIR 2006, pp. 12–17 (2006)
[4] Kiesling, K.: Metadata, metadata, everywhere - but where is the hook? OCLC Systems & Services 17, 84–88 (2001)
[5] Pitti, D.V.: Encoded Archival Description: An Introduction and Overview. D-Lib Magazine 5(11) (1999)
[6] Ponte, J.M., Croft, W.B.: A language modeling approach to information retrieval. In: SIGIR 1998, pp. 275–281. ACM, New York (1998)

[7] `http://developer.yahoo.com/yui/`

[7] Trotman, A., Sigurbjörnsson, B.: Narrowed Extended XPath I (NEXI). In: Fuhr, N., Lalmas, M., Malik, S., Szlávik, Z. (eds.) INEX 2004. LNCS, vol. 3493, pp. 16–40. Springer, Heidelberg (2005)
[8] Tsikrika, T.: Aggregation-based Semi-Structured Text Retrieval. In: Encyclopedia of Database Systems. Springer, Heidelberg (2009)
[9] Zhang, J., Fachry, K.N., Kamps, J.: Access to Archival Finding Aids: Context Matters. In: Christensen-Dalsgaard, B., Castelli, D., Ammitzbøll Jurik, B., Lippincott, J. (eds.) ECDL 2008. LNCS, vol. 5173, pp. 455–457. Springer, Heidelberg (2008)
[10] Zhang, J., Kamps, J.: Searching Archival Finding Aids: Retrieval in Original Order? In: Agosti, M., Borbinha, J., Kapidakis, S., Papatheodorou, C., Tsakonas, G. (eds.) ECDL 2009. LNCS, vol. 5714, pp. 447–450. Springer, Heidelberg (2009)