First publ. in: DocEng 2012 : proceedings of the 2012 ACM Symposium on Document Engineering; September 4 - 7, 2012, Paris, France. - New York, NY : ACM, 2012. - S. 195-204. - ISBN 978-1-4503-1116-8

A Framework for Retrieval and Annotation in Digital Humanities using XQuery Full Text and Update in BaseX

Cerstin Mahlow* Christian Grün† Alexander Holupirek† Marc H. Scholl†

*Department of German
University of Basel
4051 Basel, Switzerland
cerstin.mahlow@unibas.ch

†Database & Information Systems Group
University of Konstanz
78457 Konstanz, Germany
[firstname.lastname]@uni-konstanz.de

ABSTRACT

A key difference between traditional humanities research and the emerging field of digital humanities is that the latter aims to complement qualitative methods with quantitative data. In linguistics, this means the use of large corpora of text, which are usually annotated automatically using natural language processing tools. However, these tools do not exist for historical texts, so scholars have to work with unannotated data. We have developed a system for systematic, iterative exploration and annotation of historical text corpora, which relies on an XML database (BaseX) and in particular on the Full Text and Update facilities of XQuery.

Categories and Subject Descriptors

J.5 [**Computer Applications**]: Arts and Humanities – Linguistics; H.2.4 [**Database Management**]: Systems – Textual databases; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – Search process

Keywords

XML, TEI, database, XQuery Full Text, corpus linguistics, phraseology

1. INTRODUCTION

Traditional humanities are mainly concerned with the qualitative exploration of text to answer specific research questions. This includes investigating modern and historical texts: hand-written, printed, spoken, or in electronic form, probably mixed with images, etc. Digital humanities extend these traditional research methods and resources by applying quantitative methods to large amounts of electronically available texts. To answer linguistic questions, these methods rely on linguistically annotated texts. Usually, linguistic annotation is done automatically by applying natural language processing (NLP) tools to raw text.

However, research on historical texts in this paradigm is hampered by the fact that NLP tools are suitable for modern texts only. Historical texts reflect different spelling conventions than today, inflection might have been different, and rules for word order or for syntax might have changed over time. Additionally, some centuries ago there was no fixed set of spelling rules writers were supposed to follow—we often find different spellings for a word within one text. For these reasons, using NLP for modern language on historical texts typically yields unsatisfying results [see for example 11, 30]. Therefore, when laborious manual annotation is no option, scholars prefer to apply methods from information retrieval to find relevant information in texts.

There are currently only limited resources to support scholars from the digital humanities in searching and enriching their data. Typically, project-specific or document-specific models, methods, and tools are developed, which is not an optimal situation, as Romary [28] argues. The result of a query in corpus linguistics tools might be exported to be later evaluated manually or statistically. However, to ensure reproducibility of linguistic research and to allow for comparing variants of queries considering slightly differing perspectives, it would be necessary to annotate the original data to make found information explicit and to enrich the data with information derived from interpretation of the query and the results. The result of a query should result in an annotation layer that might be extended by manually added information. These annotation layers might then serve as a resource for higher-level investigations.

In this paper, we propose a framework supporting at the same time retrieval and annotation of linguistic structures in diachronic corpora from the digital humanities. We present the architecture of BaseX, as an instantiation of an XML database as proposed by Salminen and Tompa [29], the W3C XQuery 3.0 language, its official Full Text and Update Facility extensions, and some specifics of the implementation in BaseX. As a case in point, we give details on exploring TEI document collections of German texts from 1650 until today; we discuss the tools and resources implemented so far and demonstrate how linguists investigating diachronic phenomena benefit from the application of state-of-the-art XML technologies.

2. A FRAMEWORK FOR RETRIEVAL AND ANNOTATION OF DIACHRONIC CORPORA

At various places, large amounts of hand-written or printed texts are currently being digitized and semi-automatically converted into XML data conforming to TEI P5 [34]. These TEI-annotated corpora are important sources in the growing field of digital humanities. A typical use of such data is the exploration of the XML structure to apply different display procedures depending on user preferences. Human users, e.g., scholars, can then inspect the data, which is

tailored to their needs, or obtain a rendering that reflects the original rendering of the printed edition.

Research questions from the digital humanities may include investigations concerning the linguistic development of a certain language, exploring the ethical or legal development of societies, analyzing texts to learn about animals and plants existing at former times, getting evidence for historical events, etc. To answer these questions, researchers explore textual documents by applying queries which make use of the content and the existing annotation.

These queries typically search for concepts rather than specific words or phrases. Searching for a concept requires formulating various queries, taking into account possible variation of words to express a concept, involving variants of multi-word units like co-occurrences, collocations, or idioms. The hits are then interpreted by the researcher. In general, the automatic distinction between true and false positives is almost impossible. In most cases, corpus work involves both inspecting the text, considering already available annotation, and annotating specific aspects to be inspected in detail in a follow-up step. A corpus tool for the digital humanities thus has to support retrieval as well as annotation.

Researchers from digital humanities are linguists, historians, or jurists with limited experience in XML structures and techniques. Therefore, allowing users to query a corpus by writing XQuery expressions is no option. This also affects performance requirements. The display of results to be inspected—to decide whether to refine the query or to select certain hits for further exploration—cannot be limited to matched nodes or attributes. It involves the full rendering of hits including their context, e.g., highlighting the matched text and showing bibliographic data or preceding and following nodes. However, these additional requirements should have no considerable effect on the response times.

The corpora to be searched usually consist of various documents from different authors, written and published at different points in time, i.e., diachronic corpora consisting of heterogeneous texts. The TEI annotation of textual sources generally echoes the document structure only, there is no annotation with respect to language features. If there is linguistic annotation available—either done automatically for modern texts or manually for small document collections of historical texts—linguistic information usually is stored as separate layers to the basic data. These data can then be searched using dedicated tools; typically, special-purpose databases are used for retrieval, e.g., Corpus Workbench[1], making use of linguistic annotation like lemmatization, part-of-speech (POS) annotation, or syntactic annotation.

However, linguistically annotated corpora of a reasonable size exist for modern languages only. The use of corpus linguistic tools to explore large diachronic corpora is thus not possible and we can only make use of the TEI annotation.

2.1 State of the art and related work

Although corpus annotation is realized almost exlusively as XML annotation, there are only few NLP projects using standard XML tools or XQuery for exploration of linguistic data, see for example [12, 27, 31]. However, existing approaches have several drawbacks, due to the nature of the research objects as well as due to technical solutions chosen.

Researchers from digital humanities mainly care about the content of their textual resources. Creating a collection of relevant documents concerning a research question often results in a collection of heterogeneous documents: The documents are of different size ranging from a few kilobytes to hundreds of megabytes, eventually resulting in large collections of dozens of gigabytes; the documents might be born digital or scanned and OCR-processed; the quality of the TEI annotation depends on the annotation process used; the character encoding might be inconsistent. Therefore most projects introduce a semi-automatic preprocessing step to either harmonize documents [see for example 27] or to even get rid of annotation, because actual processing is done on raw texts [see 25]. For example *brat*, a Web-based tool for NLP-assisted annotation [32], expects input data as plain text.

Baumann et al. [2] present a system suited to handle multi-layered data, intended to overcome performance issues of other systems. However, they do not report on the *creation* of the annotation. Similarly, Eckart and Teich [14] focus on querying and representation only.

Rehm et al. [27] report response times of up to 3 hours for typical queries. With their system they are not able to achieve performance desired for interactive use. When systematically *annotating* corpora, it might be possible to split the corpus into smaller portions to be able to inspect and annotate them with a reasonable performance. However, when *searching* for evidence of concepts that might occur only rarely, splitting the corpus is no option—the researcher will lose the overview.

Annotation tools support manual annotation of small portions of the corpus to be merged later. Adding information to existing XML data is usually done by systematically inspecting all information that has already been stored, introducing new tags for marking relevant parts, or adding attributes to existing elements.

Automatic annotation of linguistic information is done by applying NLP tools. These tools have been developed for and tuned to perform best on modern newspaper texts. Texts from the 17th, 18th or 19th century differ from modern texts—i.e., late 20th or early 21st century—with respect to spelling, vocabulary, inflection, word order, syntax, or any combination of the above. Applying modern tools to old texts is thus not easily possible, as Dipper [11] or Scheible et al. [30] have demonstrated. Annotating diachronic corpora, would therefore require manual annotation, which is usually not possible. Part of an alternative solution to overcome, for example, lack of lemmatization, would be to use query expansion, i.e., generating all inflected modern and historical forms in all possible spellings for a given lemma in a query. This would, however, require generation tools that take changes in inflection paradigms and spelling variation into account; tools that currently do not exist.

2.2 An integrated system for retrieval and annotation of large XML-annotated corpora

As most of today's corpus data comes in XML-annotated format, a framework for digital humanities should make use of this annotation rather than stripping it in a preprocessing step. Additional information should be added either directly into the corpus or as stand-off markup.

XML databases suited for diachronic corpora have to meet several demands, e.g., using standards, avoiding data-model transformation, supporting stand-off annotation, providing language-sensitive full-text search, performance issues [see 13]. When choosing an XML database, we also have to consider that looking for concepts or variants of multi-word units in texts is a different task than typical data-oriented XML search. The corpus is not a collection of highly structured records, but unstructured text with some metadata. We thus need a solution that allows fast and efficient search in the content of XML nodes supported by full text indexes while also considering information stored in attributes.

Exploring a corpus in the field of digital humanities is a recursive process of applying queries to a large heterogeneous corpus, inspect-

[1] `http://cwb.sourceforge.net/`

ing results, annotating new information for some of the hits, and applying new queries making immediatly use of added annotation. A framework integrating high-performance search *and* annotation is needed. Accordingly, a separate retrieval engine such as Lucene is not the best option, while XQuery Full Text and XQuery Update [1] offer the desired facilities, as they additionally allow us to stick with the original XML format.

The integration of XQuery in a dedicated XML database like BaseX facilitates the implementation of applications that support non-technical users like linguists to query and annotate large collections of TEI documents in a comfortable environment and in real time. Using BaseX and the XQuery Full Text and Update implementation, we are able to overcome most of the deficiencies of other approaches. We can easily create corpora of heterogeneous documents; there is no need for semi-automatic preprocessing. Results of queries can be annotated immediately, this annotation can then be used in further queries without any need for additional processing steps. Retrieval and annotation can be nested, allowing scholars to refine queries and supporting interpretation of information.

3. ARCHITECTURE OF BASEX

BaseX[2] is a native XML database management system. In 2005, it was developed by the Database and Information Systems Group (DBIS) at the University of Konstanz as a research project on efficient storage layers for tree-structured data [19]. It soon turned out that the resulting storage system is suited especially well to provide visual access to large XML corpora [20]. A visual search application, which provided access to the complete catalog of the Library of the University of Konstanz, encoded as XML, has successfully spawned further research efforts during which the prototype evolved into stable software [18].

Since 2007, BaseX has been a publicly available, BSD-licensed open source project.[3] Today, the database system is actively used and developed by a growing community and a core team, which supervises the implementation efforts and introduces new features in the system. The core of BaseX is entirely written in the Java programming language. If the database is run as server, however, one of 15 client libraries for different programming languages (including C, Haskell, Perl, PHP, Ruby, and Scala) can be used to connect to the database and to process commands and queries with BaseX.

3.1 Working with BaseX

There are two main modes to operate BaseX: the standalone mode (with console or graphical user interface) and the client/server mode.

3.1.1 Standalone mode

The standalone mode is targeted at developers and XML architects who want to locally explore and work with XML data. BaseX is started as a conventional application (no configuration is needed). The standalone variant is feature complete and can be downloaded for Windows, Mac OS, and various Linux distributions from the project's website.

By default, the graphical user interface (GUI) is started, which is depicted in Figure 1. Visual access to XML data is provided by separate *views*. The figure shows the *Map View* in the bottom half. It is a space-filling representation of the opened database (`factbook.xml` in this case). The upper left pane shows the *Query Editor*, in which users can enter and evaluate XPath/XQuery expressions on the data to get immediate feedback with each key click. The upper right pane gives information about compilation, optimization, evaluation steps, and timings of the executed query. Additional hierarchical visualizations are available to explore the stored XML data. All views are tightly coupled and provide instant result feedback.

In contrast to other XML tools and editors, BaseX GUI can both store and visualize very large XML documents (files up to 421 GiB have been successfully tested[4]). Databases in BaseX are very lightweight: When XML files are opened, they are converted to database instances on the fly. This allows for a fast exploration and analysis of large XML documents and collections, since all visualizations directly interact with the underlying storage layer [17, Chapter 2].

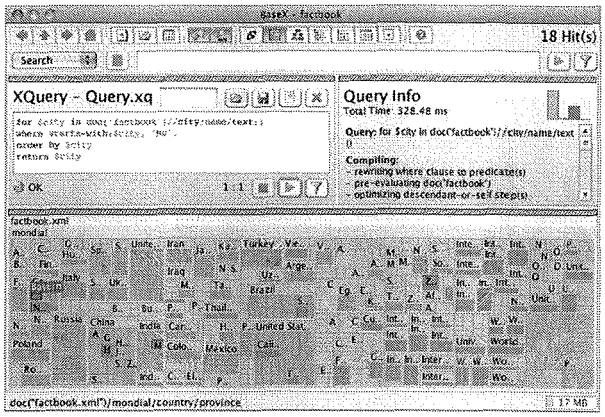

Figure 1: Standalone mode of BaseX.

3.1.2 Server mode

The server mode is the preferred solution if BaseX is expected to provide its services in a multi-user environment. BaseX Server offers a central storage for XML documents and binary files that can subsequently be accessed by remote clients in several programming languages.

XPath and XQuery are the default languages to access data stored in server database instances. This allows system architects to leverage low-level system internals in a high-level programming language. However, XQuery is far more than *just* a data query language (DQL), as described in more detail in Section 4.

In addition, BaseX offers two REST interfaces and an implementation of the WebDAV protocol for accessing and updating data. In a RESTful environment, for example, AJAX developers can send data of any format (e.g. XML, JSON, or binary data) to the server and take advantage of the XQuery language to process and retrieve relevant data.

Whatever mode is used, BaseX provides general features relevant for applications in the domain of digital humanities: (1) support for established W3C standards to operate on XML data: XQuery, XQuery Full Text, and XQuery Update, (2) a high compliance level regarding official test suites[5], (3) supporting infrastructure for large textual corpora: Text, Attribute, Full-text and Path indexes, and (4) facilities for tuning, optimizing, and accessing indexes.

[2] `http://basex.org`

[3] Source code can be retrieved for further development, adaptation, and improvements at `https://github.com/BaseXdb` and `http://basex.org/open-source/`

[4] `http://docs.basex.org/wiki/Statistics`

[5] `http://dev.w3.org/2006/xquery-test-suite/PublicPagesStagingArea/XQTSReportSimple_XQTS_1_0_2.html`

3.2 Building applications

At least three different ways exist to build higher-level applications with BaseX: (1) Since the database engine does not depend on other libraries and has a small memory footprint, BaseX can be a good choice for *embedded* systems. It can be used as an XML storage library and/or XQuery processor inside a Java application. (2) BaseX can be used as a classic *client/server* architecture, as described in Section 3.1. (3) The system can be deployed as a *web application* as central part of a pure *X-technology stack*.

XML, XQuery and XHTML are an ideal match to present and process information resources in a platform-neutral way. Whenever the underlying datasets are originally stored in XML, or can be easily represented as such, XQuery is the domain-specific processing language to *filter, select, search, join, sort, group, aggregate, transform, and restructure*, in short, analyze and process, stored data.

BaseX provides a service infrastructure to implement and deploy XQuery-based web applications. That way, XML technology can be applied on all layers of a classical three-tier-architecture [23]. The *persistence* layer is provided by a native XML database, *business logic* is implemented in XQuery, and the *presentation* layer is primarily driven by XHTML.

As such, a single data model is used throughout the architecture and no conversions have to be applied between the layers. If the "unified technology stack" is used, as shown in Figure 2, the full potential of the W3C language family can be leveraged, and benefits can be expected in terms of (1) a lean system architectures, with less components involved, and (2) a reduced amount of code, as no glue code is needed.

The architecture allows for the development of applications solely relying on the W3C technology family: Applications are provided with a uniform search and retrieval service, and can now be implemented on a more high-level and generic abstraction layer, while still being backed by a full-fledged database support.

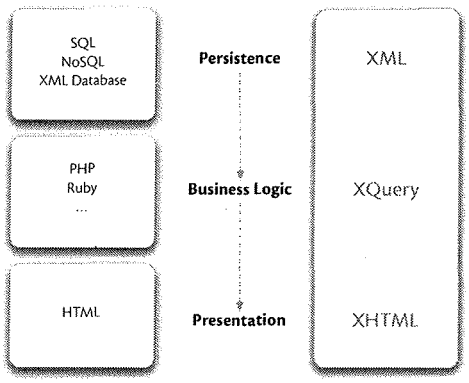

Figure 2: Uniform Application Stack: XML technology on all three tiers of a system architecture

4. XQUERY AND ITS FULL TEXT AND UPDATE EXTENSIONS

While XQuery is often labeled as query language for XML data, and put on a same level with SQL, it is actually a full, Turing-complete, functional programming language, which makes it perfectly suited for representing and processing full information workflows with XML data. It is continuously developed and enhanced with new extensions: currently, XQuery 3.0 is being standardized, and full-text, update and scripting extensions provide additional features for information retrieval, modifications, and batch jobs.

```
//library/title[content contains text
  ("apple" ftor "pear") ftand ("stem" ftor "tree")
  using diacritics insensitive
  using thesaurus default
  using case insensitive
  using language "en"
  using stemming
  ordered distance at most 5 words]
```

Listing 1: Example XQuery expression

The XQuery core language makes use of XPath to navigate through XML nodes (elements, attributes, texts, etc.) in a single document or a collection of documents. So-called FLWOR expressions can be used to loop through, filter and order XQuery items. New XML nodes can be constructed on the fly, which can then be handled like existing XML nodes. The language provides support for user-defined functions and modules in different namespaces. Due to its functional nature, XQuery has only expressions and no statements: expressions within a function body are evaluated and returned as values. The data model of XQuery treats all values as sequences. Sequences are lists of items, and items can be nodes or atomic values. Atomic values may be for example strings, doubles, integers, booleans, or dates [4]. Nested sequences will be automatically flattened.

With XQuery 3.0, many important features have been added, which make the language more suitable for universal processing of XML encoded information: function items (also known as lambdas) can be used to write more modular code and use XQuery as a fully functional language. In line with the typed lambda calculus, higher-order functions can now be passed as arguments to or returned as results from functions. Next, the new language version introduces a try/catch expression to handle errors at runtime; serialization parameters can be specified within the query, the FLWOR expression has been extended with a `group by` clause, all its clauses can now be placed in an arbitrary order, and annotations can be used to change the visibility of functions or assign them special, implementation-specific properties.

In SQL, many vendor-specific extensions exist to process full-text requests. An alternative, unified approach has been taken in the XML domain: The XPath/XQuery Full Text Recommendation of the W3C [1] is fully composable and tightly coupled with the core language. Since its finalization, it is continuously attracting more and more users and developers from the information retrieval community. The recommendation offers a wide range of content-based query operations, classical retrieval tools such as stemming and thesaurus support, and an implementation-defined scoring model that allows core developers to adapt their database to a large variety of use-cases and scenarios. BaseX was the first implementation to fully support all features of the new specification; Qizx [16] and MXQuery [15] are two other implementations that are available at the time of writing.

The syntax of a simple full-text expression is similar to a "general comparison" in XQuery [5]. A `contains text` expression can get pretty verbose: the right hand side can be extended by numerous logical connectives, match options, and positional filters as shown in Listing 1.

BaseX provides an additional, implementation-specific match option `fuzzy`, which is based on an optimized variant of the Levenshtein algorithm [33]. Depending on the length of a string, a certain

```
declare namespace output =
'http://www.w3.org/2010/xslt-xquery-serialization';
declare option output:method 'xhtml';
declare option output:omit-xml-declaration 'no';
declare option output:doctype-public
    '-//W3C//DTD HTML 4.01 Transitional//EN';
declare option output:doctype-system
    'http://www.w3.org/TR/html4/loose.dtd';
declare variable $words external := '-';
<html>
  <head>
    <title>Search: { $words }</title>
  </head>
  <body>{
    for $m in doc('library')//medium
    where $m/content contains text { $words }
    return (
      <h1>{ $m/title/data() }</h1>,
      <div>{ $m/content }</div>
    )
  }</body>
</html>
```

Listing 2: XQuery expression yielding an XHTML document

number of deviations from the search string will be ignored. A deviation may either be a missing, additional, wrong, or transposed character. The following query contains a single transposition and yields true:

```
'apple' contains text 'appel' using fuzzy
```

A further noteworthy enhancement, which is helpful for the discussed use case, and likely to be included in a future version of the Recommendation [6], is the possibility of highlighting found tokens in the query results with the `ft:mark()` function. With `ft:extract()`, longer texts can be shortened and limited to the regions that contain the relevant keywords. More details on the low-level implementation of XQuery Full Text in BaseX are described by Grün et al. [21].

Another essential requirement for query languages is the possibility of performing updates. The official XQuery Update Facility [8] fills this gap by introducing four new expressions to insert new data and modify or delete existing data (`insert`, `replace`, `rename`, and `delete`), thus offering a data manipulation language (DML). An additional `transform` expression allows developers to modify nodes in main memory. A special characteristic of XQuery Update is that all atomic update operations are first moved to a *pending update list*, which is processed in batch mode after the query itself has been evaluated. Next, similar to SQL, updating XQuery expressions may not return any results. Due to these two constraints, and numerous validity checks, any side effects are avoided: all updates are guaranteed to be atomic, and no data can be returned that has previously been deleted.

From the implementor's view, the batch processing allows for additional optimizations, as none of the reading data references need to be preserved at the final stage of database modification. An obvious drawback of this design approach is that subsequent read and write operations need to be encapsulated in multiple queries.

With XQuery 3.0 and its extensions, there is no need any more to mix different technologies and data models, e.g., MySQL, PHP, and HTML: complete web pages can be created without switching the language and platform. As a result, application development is getting simplified, and performance is improved as no abstraction layers need to be passed, as indicated in Section 3.2. Listing 2 is an example for an XQuery 3.0 expression that yields a valid XHTML document.

Especially XQuery's Full Text extension makes XML database systems like BaseX an interesting choice for building information retrieval systems in Digital Humanities, as we elaborate in Section 5. To offer high-performance throughput of XQuery Full Text queries, BaseX implements Text, Attribute, Full-Text and Path-Summary indexes to speed up the evaluation process. The index structures are designed to support more than 20 languages and incorporate features, as wildcards, stemming, case sensitivity, diacritics, TF/IDF scoring, and stop words. A point worth mentioning is the ability to programmatically access database internals, such as values stored in indexes from within XQuery.

5. USE CASE: RETRIEVING AND ANNOTATING IDIOMATIC PHRASES

In the SNSF-funded project "German Proverbs and idioms in language change. Online dictionary for diachronic phraseology (OLdPhras)"[6], we are interested in finding historical evidence for phrasemes—in linguistics sometimes also referred to as phraseological units, idioms, or set phrases—in German texts from 1650 until today. A list derived from phraseme collections and general-purpose dictionaries from the 18th to the 21st century comprises the inventory of the intended dictionary.

More abstractly, phrasemes could be considered non-Fregian discontinuous multi-word expressions within sentence boundaries, where the meaning of the whole unit cannot be deduced from the meaning of the parts, see examples 1a and 2a below. Diachronic change of phrasemes might occur on various levels: lexical units, syntactic structure of the phraseme, meaning, syntactic role of the phraseme, etc., making searching for instances of phrasemes a complex task. Retrieving and annotating phrasemes is one step in the creation of the online dictionary.

Resources are a collection of diachronic TEI-annotated texts and a collection of printed dictionaries representing the knowledge about German phrasemes at different points in time from the 18th to the 21st century. The first step is the compilation of relevant data. Only some of the dictionaries are available in electronic form, i.e., have been digitized. Relevant information from dictionaries will be extracted semi-automatically or manually and stored using BaseX in XML format for further use. Relevant information from texts, i.e., evidence of phrasemes, will be collected and annotated within the database. Storing this information basically means storing annotation layers, not extracts from the texts.

In a further step, BaseX is used for the creation of single dictionary entries: all information concerning a particular phraseme—i.e., extracted parts of other dictionaries and annotated evidence— is merged. The phraseologist will arrange these elements and write comments concerning diachronic change. The resulting XML documents will then form the actual OLdPhras dictionary. Additionally, the annotation layers covering evidence of phrasemes can be merged with the TEI-annotated original texts, resulting in a corpus annotated with information on phrasemes. Figure 3 presents the overall architecture of the OLdPhras system. For more information about the project in general see [22, 24].

Phrasemes are relatively rare in corpora of written language; large corpora are thus required to obtain a significant amount of evidence [9, 10]. As diachronic corpora we use the *Deutsches Textarchiv* (DTA)[7], the *TextGrid Digitale Bibliothek* (DB125)[8], and *GerManC* [3], comprising digitized historical German texts up to the

[6] `http://oldphras.net`
[7] `http://deutschestextarchiv.de`
[8] `http://www.textgrid.de/digitale-bibliothek.html`

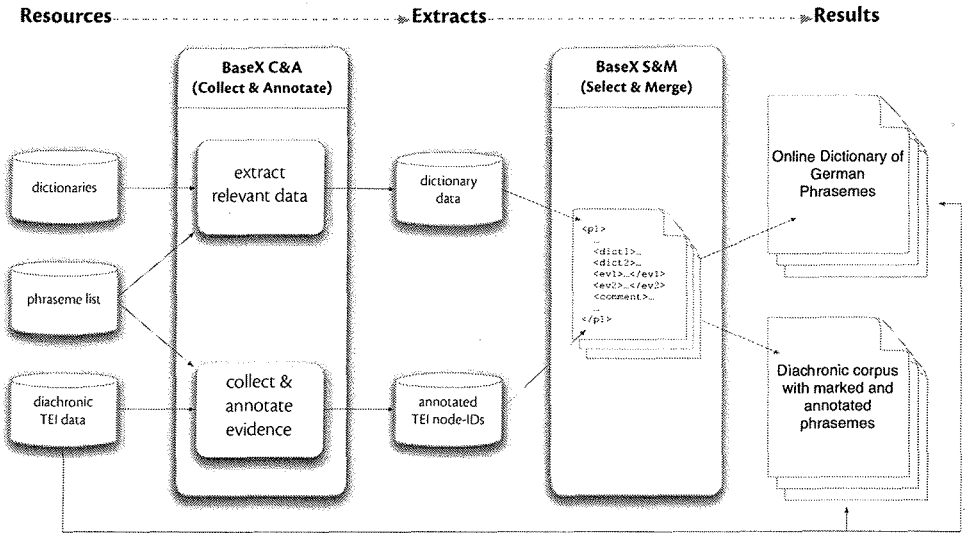

Figure 3: Overall architecture of the system.

early 20th century. All corpora include TEI-annotated documents, but the annotation and the organization of the documents differ.

In the rest of this section, we first outline general challenges for retrieving and annotating phrasemes and then present the principles of our solution based on XQuery 3.0 and its Full Text and Update extensions, and the implementation using BaseX.

5.1 Challenges

5.1.1 Linguistic challenges

Typically, multi-word units like co-occurrences, collocations, or phrasemes are searched for using dedicated tools from corpus linguistics, e.g., Corpus Workbench, making use of pre-existing linguistic annotation. Queries can then be constructed using the base forms of words, specifying syntactic relations, and allowing for instantiations of a particular POS. For example, query 1b would search for phrases consisting of a preposition, followed by a determiner, followed by any inflected form of *Strom* 'current', followed by any inflected form of the verbs *schwimmen* 'to swim' or *treiben* 'to drift'. Between the noun and the verb there might be other words, e.g., adverbs. With this query, we could look for evidence of variants of phraseme 1a.

(1) a. *gegen den Strom schwimmen*
'to swim against the current'
(today's meaning 'to act different than the majority')

b. `PREP DET Strom * schwimmen|treiben`

In general, looking for phrasemes in texts with a focus on diachronic change concerning meaning, vocabulary, and structure, is a complex task. The starting point is a modern base form as listed in dedicated phraseme collections or general purpose dictionaries, like examples 1a and 2a: the verb is in infinitive form at the very last position as in 1a, valency fillers of the verb might be instantiated as in 2a or not mentioned at all as in 1a (a subject is needed, someone able to swim). From this base form, a linguist needs to construct a query that allows for variation on all elements, i.e., meaning, vocabulary, and structure, either one at a time or in combination.

For example, for phraseme 2a, on the lexical level we could expect variation with respect to the actual fruit—e.g., pears or plums instead of apples— or with respect to the part of the tree—e.g., the tree as such, a branch, or a different plant. The best query would thus be query 2b, in which the first element—the valency filler of *to fall from*—is a fruit, and the second a plant or a part of a plant. Both valency fillers may be single nouns, complex noun phrases, or pronouns denoting real-world objects. And of course variation in word order is possible, too.

However, our corpus does not provide linguistic annotation. Even when using linguistically deeply annotated corpora and semantic resources like GermaNet[9], such queries would be hard to construct and it is almost impossible to find, for example, evidence 2c in a text from 1669 [26].

(2) a. *der Apfel fällt nicht weit vom Stamm*
'like father like son'
(literally 'the apple does not fall far from the stem')

b. `something`$_1$ `fall from something`$_2$

c. *die birn nit wey vom baum falt*
(literally 'the pear not far from tree falls')

The best way to look for a variant of an idiomatic phrase is thus to look for some core elements, like *fall from*, and then to inspect the results manually to distinguish true from false positive hits. For this purpose, string-based queries are sufficient. Variation of a search string would include spelling variation and inflection. Therefore, XQuery Full Text with fuzzy search (for spelling variation) and stemming (for handling basic inflectional variation) is currently the most appropriate and flexible solution we are aware of.

5.1.2 Technical challenges

In principle, all XML documents represent scanned, OCR-processed, manually corrected, and semi-automatically TEI-annotated texts. However, documents are of different size, ranging from 30 KB to 90 MB. The collection of all XML documents is 2.9 GiB in size. The documents are organized differently; they may contain a single text or only part it written by a single author, a compilation of dozens of poems by a single author, or all books (which in turn can

[9] `http://www.sfs.uni-tuebingen.de/lsd/index.shtml`

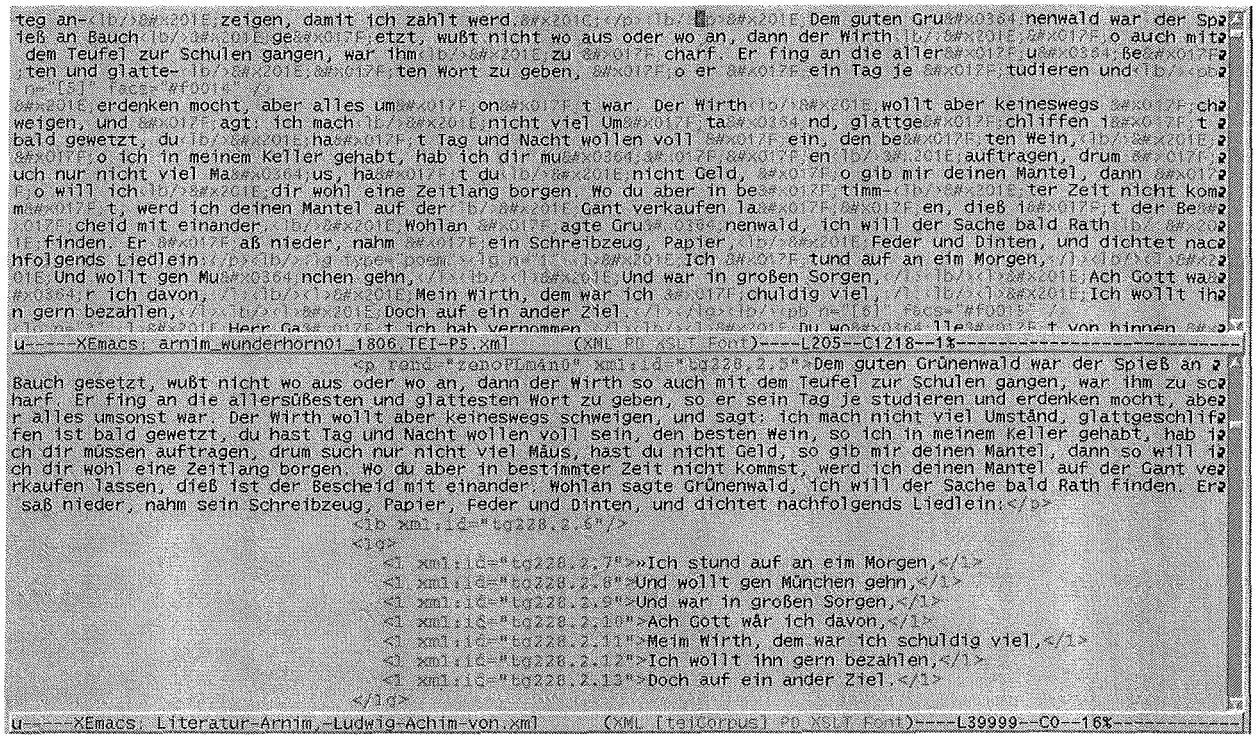

Figure 4: Excerpt from XML representation of "Des Knaben Wunderhorn. Alte deutsche Lieder" by Achim von Arnim and Clemens Brentano

be compilations of poems or tales, single novels, etc.), of a single author. The texts themselves are also not alike, as the documents represent different literary genres, which differ from each other in style and syntax. In addition, they have often been printed by different publishers at different points in times, resulting in different layouts or document structures.

Moreover, the digitization guidelines differ as well: DTA and GerManC aim to digitize first editions, resulting in texts in the "original" language from the time of the writing or the first printing of a text. DB125 often used later editions, resulting in rather modern language even for texts originally written in the 17th century. The TEI annotation of DTA closely mirrors the original format and therefore includes page breaks and line breaks. The latter preserves hyphenation, which makes searching for words difficult. Applying appropriate rendering scripts to this data would easily allow displaying the text as lookalike to the printed source. The TEI annotation of DB125, on the other hand, does not preserve line breaks (except for poems) and thus no hyphenation occurs.

Figure 4 contains the same excerpt from "Des Knaben Wunderhorn. Alte deutsche Lieder" by Achim von Arnim and Clemens Brentano as found in DTA (top buffer) and in DB125 (lower buffer). It illustrates the characteristics of the two XML formats described above. Some texts have been included in both corpora: in this case, DB125 used an edition from 1979, whereas the source in DTA was printed in 1806. As BaseX allows the creation of collections with XML documents of different formats and different DTDs or XML Schemas, or no schemas at all, it is an optimal solution to store and query our heterogeneous corpus.

The smallest structural units are paragraphs (for prose) and lines (for poems or dramas). Within a paragraph or line, there might be rendering information (to preserve special formatting of the resource) and page breaks or line breaks. The text we hope to find phrasemes in is the content of these paragraph elements (<p>) and line elements (<l>).

A further challenge is the total amount of data to be processed. For phraseological research, the available data might be always too limited while, at the same time, it will be too large to be efficiently handled with common databases. Exploring texts for evidence of phrasemes means constructing the best query in a recursive process by querying the corpus, inspecting results, and refining the query. The phraseologists might also wish to save intermediate results as reference point for constructing optimized queries or to compile a result collection when no single best query is possible. Both scenarios require fast execution of the query and fast display of the result.

A pilot study revealed that for some phrasemes there is only little evidence, i.e., less than five relevant hits, but for others there are more than 800 relevant hits. On average, we have 90 relevant hits per phraseme and around 200 irrelevant hits. For a particular query there might be no hit at all or up to several thousand. Given the goal of investigating 1,000 phrasemes, this not only takes a lot of time for query formulation, result inspection, and further annotation, but it also creates a lot of data.

5.2 Applying XQuery expressions to the corpus

In our context, the main focus while searching the XML documents is not on the *structure*, but on the *contents*: The structure is used only to determine the bibliographic data needed to identify relevant excerpts and to be able to display context necessary for

annotating linguistic aspects like meaning or register. Bibliographic information can be accessed by traveling up the document tree. The original page on which the text would be found in the printed version can be accessed by descending to the next page-break information. Context is defined as a certain number of preceding or following text nodes.

Index structures are mandatory if large databases are to be queried. For our purpose, we create a full-text index, including all stop words. Usually, stop words are conjunctions, prepositions, negation elements, pronouns, auxiliary verbs, etc. However, these are essential for retrieving phrasemes as they probably belong to the core of a phraseme. It might be better to search for combinations of prepositions and verbs without specifying nouns when looking for variation of nouns in a phraseme like 2a.

The creation of the full-text index results in a 4 GiB database. The index is applied if exact or fuzzy matches are performed. A second full-text index is created, in which all tokens are stemmed, and the resulting index is used to query inflected word forms. The German stemming algorithm is based on Caumanns [7].

A particular multi-word unit might be used literally or idiomatically—the latter constitutes a phraseme. However, we are interested in both occurrences in order to make statements about change: when did a multi-word unit become a phraseme and was less used literally or vice versa; was there a shift in meaning? Therefore, scoring or ranking of hits is not useful for this application; we are interested in all hits. As multi-word units might occur in various word order and be discontinuous, we cannot automatically distinguish correct and false positive hits, the latter defined as occurrence of all elements of the query but with no syntactic or semantic relation. The distinction has to be made by the phraseologist, i.e., manually.

With XQuery Full Text, queries can be created that specify (1) whether the elements of a search string have to occur as continuous string, i.e., as a "phrase", (2) as discontinuous string, i.e., all words have to occur or only some of them, (3) preserving or not the given order, or (4) within a given window. The latter allows for mimicking searching within sentence boundaries.

Listing 3 shows XQuery expressions from the query logs, which are related to find evidence of phraseme 2a. The first query contains no stemming option, but looks for exact matches of the search strings only. The other queries include the preposition *vom*, use different distances, and enforce the search strings to occur in the given order. All elements of the search string have to occur within a single paragraph or a line.

The results of a query, manually selected for annotation, are stored in a separate database, the *collect* database. The XQuery expression created by the phraseologist is stored as well. This serves documentation purposes, checking and approving procedures—this applies also to the information on who selected a specific hit—, and also allows for the creation of related or "better" queries by serving as inspiration. Listing 4 shows the structure of those entries.

The phraseologist will create various queries to find evidence for one phraseme as shown in Listing 3. After executing a query, she selects hits—i.e., paragraphs or lines where the search string was found in—to be annotated and saves the results. Hits not selected will be stored as well, having the value no for the element `<selected>` (also for documentation purposes and to be able to later automatically determine the "best" query with respect to recall and precision). The `<node>` element references the node-ID of the hit. When executing another query for the same phraseme, the results might include nodes that are already matched by a previous query and already stored as evidence for the particular phraseme. These nodes are no longer selectable, but the current query will be

```
//*[text() contains text
  ('Apfel' ftand 'Stamm' ftand 'fällt')
  distance at most 10 words][self::*:p or self::*:l]

//*[text() contains text
  ('vom' ftand 'stamm')
  using stemming using language "de"
  distance at most 10 words ordered]
  [self::*:p or self::*:l]

//*[text() contains text
('vom' ftand 'stamm')
  using stemming using language "de"
  distance at most 2 words ordered]
  [self::*:p or self::*:l]

//*[text() contains text
  ('vom' ftand 'stamm')
  using stemming using language "de"
  distance at most 1 words ordered]
  [self::*:p or self::*:l]
```

Listing 3: XQuery expressions intended to find evidence of the phraseme *Der Apfel fällt nicht weit vom Stamm*

```
<entry time = "2012-03-29T17:43:29" user = "..." >
   <node>3438425</node>
   <phraseme>Ad0018</phraseme>
   <query>[text() contains ...]</query>
   <secondquery>[text () contains ...]</secondquery>
   <selected>yes</selected>
</entry>
```

Listing 4: Pseudo entry in the *collect* database

stored in a `<secondquery>` element in the entry that represents this evidence.

We follow the paradigm of multiple layers to annotate the corpus. One layer contains information about which phraseme was found in a particular text node. The node-ID of this particular paragraph or line in the XML documents collection is used as reference point. The phraseme is also stored as a reference (`<phraseme>`) pointing to a list of given phrasemes and their prototypical meaning as found in dedicated collections or dictionaries. If an evidence is annotated, attributes containing information about linguistic aspects are created—i.e., register, modality, negation, voice, etc.—and stored in the *annotation* database, which contain entries like the ones shown in Listing 5. A particular paragraph or line may contain more than one phraseme.

As briefly described above, collecting and annotating evidence re-

```
<node id = "3438425" >
   <phraseme
      id = "Ad0018"
      mark = "Apfel fällt selten weit vom Stamme"
      voice = "active"
      negation = "no"
      meaning = "idiomatic"
      register = "..."
      time = "2012-04-19T16:04:33"
      user = "..."
   />
   <phraseme
      id = "Ad0048"
      mark ...
   />
</node>
```

Listing 5: Pseudo entry in the *annotation* database

sults in the creation of several sub-databases for intermediate results, to be later integrated into or merged with the original database. With XQuery Update, the original node can be replaced by the annotated one, in order to show evidence of phrasemes and their linguistic features within the original text.

5.3 Web interfaces for collecting and annotating evidence

The BaseX GUI is used for cleanup of annotation data whenever the phraseologists decide to no longer annotate specific features or to have additional annotation with default values. The GUI is also used on a local machine for developing efficient XQuery expressions to be used in the Web interface.

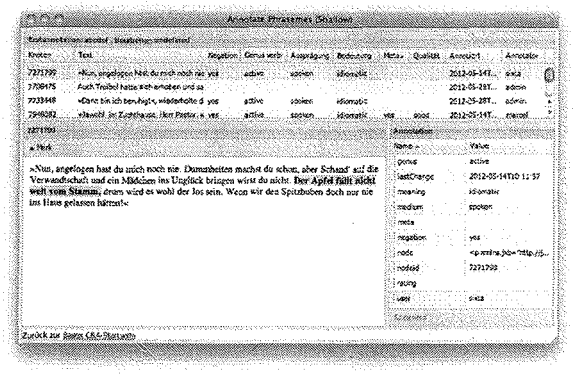

Figure 5: User interface for annotating phrasemes

For actual retrieval and annotation, the BaseX client/server architecture is used to allow remote and concurrent access for multiple users. The user interface for the phraseologists reflects the two-step process, consisting of collecting and annotating evidence: one interface allows retrieving evidence and storing relevant nodes, the other interface, as shown in Figure 5, supports annotation and is preferably used after collecting evidence for a particular phraseme is finished. The interfaces are implemented with Ext JS 4[10], and the communication with the BaseX server is based on the Perl API. Perl allows efficient and comfortable manipulation of the XQuery results in order to be displayed in a form convenient for phraseologists. The users of the web front-end do not have to write XQuery expressions, but they are provided with a simplified query language, in which most of the elements are added by choosing from fixed options. The input is then transformed into the final XQuery expression.

6. CONCLUSION

In this paper, we have given an insight into the current state of the art of XQuery, as implemented in the native XML database BaseX.

In contrast to Eckart [12, p. 187], who concluded that XML technologies are not appropriate to handle linguistic data, we could show that the capabilities of XQuery Full Text and Update now enable linguistically motivated exploration of heterogeneous documents. As a case in point, we presented the development of a framework for retrieval and annotation of phrasemes in diachronic texts.

This approach is transferable to other linguistic research questions investigating semi-structured data. An XML database like BaseX is more suitable than a relational database. XQuery 3.0 and its Full Text and Update extensions are the basis for managing, retrieving, and annotating data consistently and efficiently in a single framework.

The search and annotation interface is put into productive use. The phraseologists in the OLdPhras project will create an evidence resource to serve as one source for the edition of dictionary entries. With our implementation, we proved that project or data specific development of XML dialects for storing, retrieving, and annotating as usual today [35] is not necessary. The capabilities of XQuery are, in our experience, powerful enough to effectively meet the requirements of corpus linguistics.

[10] `http://www.sencha.com/products/extjs`

7. ACKNOWLEDGEMENTS

Alexander Holupirek is supported by the DFG Research Training Group GK-1042 *Explorative Analysis and Visualization of Large Information Spaces*. Research on phrasemes is funded by the SNSF under grant number 129577.

References

[1] S. Amer-Yahia et al. XQuery and XPath Full Text 1.0. W3C Candidate Recommendation. `http://www.w3.org/TR/xpath-full-text-10`, May 2008.

[2] S. Baumann, C. Brinckmann, S. Hansen-Schirra, G.-J. Kruijff, I. Kruijff-Korbayová, S. Neumann, and E. Teich. Multi-dimensional annotation of linguistic corpora for investigating information structure. In A. Meyers, editor, *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*, Stroudsburg, PA, USA, May 2004. Association for Computational Linguistics.

[3] P. Bennett, M. Durrell, S. Scheible, and R. J. Whitt. Annotating a historical corpus of German: A case study. In *Proceedings of the LREC 2010 Workshop on Language Resource and Language Technology: Standards - state of the art, emerging needs, and future developments*, pages 64–68, Paris, 2010. ELRA.

[4] A. Berglund et al. XQuery 1.0 and XPath 2.0 Data Model. `http://www.w3.org/TR/xpath-datamodel/`, December 2010.

[5] S. Boag et al. XQuery 1.0: An XML Query Language. W3C Recommendation. `http://www.w3.org/TR/xquery`, January 2007.

[6] P. Case. XQuery and XPath Full Text 3.0 Requirements and Use Cases. `http://www.w3.org/TR/xpath-full-text-30-requirements-use-cases/`, March 2012.

[7] J. Caumanns. A fast and simple stemming algorithm for german words. Technical report, Freie Universität Berlin, Fachbereich Mathematik und Informatik, 1999.

[8] D. Chamberlin, M. Dyck, D. Florescu, J. Melton, J. Robie, and J. Siméon. XQuery Update Facility. `http://www.w3.org/TR/xquery-update-10/`, March 2011.

[9] J.-P. Colson. The World Wide Web as a corpus for set phrases. In H. Burger, D. Dobrovol'skij, P. Kühn, and N. R. Norrick, editors, *Phraseology*, Handbooks of Linguistics and Communication Science, pages 1071–1077. Walter de Gruyter, Berlin/New York, 2007.

[10] A. P. Cowie. Phraseology and corpora: some implications for dictionary-making. *Lexicography*, 12(4):307–323, 1999.

[11] S. Dipper. POS-tagging of historical language data: First experiments. In M. Pinkal, I. Rehbein, S. Schulte im Walde, and A. Storrer, editors, *Semantic Approaches in Natural Language Processing: Proceedings of the Conference on Natural Language Processing 2010 (KONVENS)*, pages 117–121, Saarbrücken, Germany, 2010. Universaar.

[12] R. Eckart. Towards a modular data model for multi-layer annotated corpora. In *Proceedings of the COLING/ACL on Main conference poster sessions*, COLING-ACL '06, pages 183–190, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[13] R. Eckart. Choosing an XML database for linguistically annotated corpora. *Sprache und Datenverarbeitung*, 32(1), 2008.

[14] R. Eckart and E. Teich. An XML-based data model for flexible representation and query of linguistically interpreted corpora. In *Data Structures for Linguistic Resources and Applications - Proceedings of the Biannual Conference of the Society for Computational Linguistics and Language Technology (GLDV)*, pages 327–336, 2007.

[15] P. Fischer et al. MXQuery – a low-footprint, extensible XQuery Engine. `http://www.mxquery.org`, 2009.

[16] X. Franc. Qizx/db. `http://www.xmlmind.com/qizx/`, 2012.

[17] C. Grün. *Storing and Querying Large XML Instances*. PhD thesis, Universität Konstanz, Konstanz, 2011.

[18] C. Grün, J. Gerken, H.-C. Jetter, W. A. König, and H. Reiterer. MedioVis – A User-Centred Library Metadata Browser. In A. Rauber, S. Christodoulakis, and A. M. Tjoa, editors, *ECDL*, volume 3652 of *Lecture Notes in Computer Science*, pages 174–185. Springer, 2005.

[19] C. Grün, A. Holupirek, M. Kramis, M. H. Scholl, and M. Waldvogel. Pushing XPath Accelerator to its Limits. In *ExpDB*, 2006.

[20] C. Grün, A. Holupirek, and M. H. Scholl. Visually exploring and querying XML with BaseX. In A. Kemper, H. Schöning, T. Rose, M. Jarke, T. Seidl, C. Quix, and C. Brochhaus, editors, *BTW*, volume 103 of *LNI*, pages 629–632. GI, 2007.

[21] C. Grün, S. Gath, A. Holupirek, and M. H. Scholl. XQuery full text implementation in BaseX. In *Proceedings of the 6th International XML Database Symposium on Database and XML Technologies*, pages 114–128, Berlin, Heidelberg, 2009. Springer.

[22] B. Juska-Bacher and C. Mahlow. Phraseological change – a book with seven seals? Tracing diachronic development of German proverbs and idioms. In P. Bennett, M. Durell, S. Scheible, and R. J. Whitt, editors, *New Methods in Historical Corpus Linguistics*, volume 3 of *Corpus linguistics and Interdisciplinary perspectives on language*. Gunter Narr, Tübingen, Germany, 2012.

[23] M. Kaufmann and D. Kossmann. Developing an Enterprise Web Application in XQuery. In *ICWE*, pages 465–468, 2009.

[24] C. Mahlow and B. Juska-Bacher. Exploring New High German texts for evidence of phrasemes. *Journal for Language Technology and Computational Linguistics*, 26(2): 117–128, 2011.

[25] M. Poesio, E. Barbu, E. Stemle, and C. Girardi. Structure-Preserving Pipelines for Digital Libraries. In *Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH 2011)*, pages 54–62, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[26] Rechtsquellenstiftung des Schweizerischen Juristenverbandes, editor. *Appenzeller Landbücher*, volume SSRQ AR/AI 1 of *Sammlung Schweizerischer Rechtsquellen*. Schwabe, Basel, Switzerland, 2009.

[27] G. Rehm, R. Eckart, C. Chiarcos, and J. Dellert. Ontology-based XQuery'ing of XML-encoded language resources on multiple annotation layers. In N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, and D. Tapias, editors, *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Paris, May 2008. ELRA.

[28] L. Romary. Stabilising knowledge through standards: A perspective for the humanities. In K. Grandin, editor, *Going Digital. Evolutionary and Revolutionary Aspects of Digitization*, volume 147 of *Nobel Symposium*, pages 188–218. Science History Publications, New York, NY, USA, 2011.

[29] A. Salminen and F. W. Tompa. Requirements for XML document database systems. In *Proceedings of the 2001 ACM Symposium on Document engineering*, DocEng '01, pages 85–94, New York, NY, USA, 2001. ACM.

[30] S. Scheible, R. J. Whitt, M. Durrell, and P. Bennett. Evaluating an 'off-the-shelf' POS-tagger on Early Modern German text. In *Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH 2011)*, pages 19–23, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[31] S. H. Schirra, S. Neumann, and M. Vela. Multi-dimensional annotation and alignment in an English-German translation corpus. In *Proceedings of the 5th Workshop on NLP and XML: Multi-Dimensional Markup in Natural Language Processing*, NLPXML '06, pages 35–42, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[32] P. Stenetorp, S. Pyysalo, G. Topić, T. Ohta, S. Ananiadou, and J. Tsujii. brat: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, Stroudsburg, PA, USA, Apr. 2012. Association for Computational Linguistics.

[33] E. Ukkonen. Algorithms for Approximate String Matching. *Information and Control*, 64(1-3):100–118, 1985.

[34] C. Wittern, A. Ciula, and C. Tuohy. The making of TEI P5. *Literary and Linguistic Computing*, 24(3):281–296, May 2009.

[35] A. Zeldes, J. Ritz, A. Lüdeling, and C. Chiarcos. ANNIS: A search tool for multi-layer annotated corpora. In *Proceedings of Corpus Linguistics 2009, Liverpool, July 20-23*, 2009.