An Overview of Canonical Text Services

Jochen Tiepmar*, Gerhard Heyer

Natural Language Processing Group, University of Leipzig, Leipzig, 04009, Saxony, Germany

Copyright ©2017 by authors, all rights reserved. Authors agree that this article remains permanently open access under the terms of the Creative Commons Attribution License 4.0 International License

Abstract This paper provides a comprehensive overview of Canonical Text Services (CTS) and the surrounding tools that were developed on the basis of a MySQL based implementation. As such it covers a broad set of topics including a general explanation of CTS, various software tools and a wide array of text mining techniques. The goal is to compile the relatively widespread and potentially confusing amount of information into one document that focuses on the practical aspects and implications for researchers that work with text data. More technically focused aspects are discussed in the two papers that accompany this implementation ([20] and [21]) and the official CTS specifications¹. Additionally this paper introduces a licensing mechanism, a CTS based citation analysis workflow, a real time text alignment method and set of management tools including a central namespace resolver for CTS URNs.

Keywords Canonical Text Service, Linguistic Infrastructures, Webservice, Text Data, Text Mining

1 Introduction

Computer Science and Humanities so far have acted in their working methodologies more as antipodes rather than focusing on the potential synergies. However, recent advances in digitizing historical texts, and the search and text mining technologies for processing these data indicate an area of overlap that bears great potential. For the Humanities the use of computer based methods may lead to more efficient research (where possible) and the raising of new questions that without such methods could not have been dealt with. For Computer Science turning towards the Humanities as an area of application may pose new technical problems that also lead to rethinking present approaches hitherto favoured by Computer Science and developing new solutions. To the extent that applications of Computer Science have always lead to a replacement of analogue by digital media and pro-

cesses, digital media and processing models have an increasing impact also on traditional work flows based on analogue media in the Humanities and Social Sciences. By focusing on text as the main data type in the text oriented Humanities we can highlight the benefit that can be gained from the combination of digital document collections and new analysis tools in Computer Science. Thereby all kinds of humanities that work with historical or present day texts and documents are enabled to ask completely new questions and deal with text in a new manner. In detail, these methods concern, amongst others,

- The *qualitative* improvement of the digital sources (standardization of spelling and spelling correction, unambiguous identification of authors and sources, marking of quotes and references, temporal classification of texts, etc.);
- The *quantity* and structure of sources that can be processed (processing of very large amounts of text, structuring by time, place, authors, contents and topics, comments from colleagues and other editions, etc.);
- The *kind* and quality of the analysis (big data driven studies, strict bottom-up approach by using text mining tools, integration of community networking approaches, etc.).

However, in order to also guarantee reuse of data and software, reference collections of digital text are required that are widely accepted in the scientific community. These reference collections should be *language independent*, and provide *persistent* and *citable* reference data with a high degree of *scalable granularity*, access to the data should be *intuitive* and easy to learn. In what follows we shall present an implementation of Canonical Text Services (CTS) as an ecosystem for the Digital Humanities that serves to standardize reference collections of digital text and enables easy access and processing of text data. We begin by presenting the CTS specification followed by a detailed discussion of how CTS helps to meet some general requirements on reference collections of digital text data. We then present some examples of reference collections, such as Deutsches Textarchiv, and finally sketch some applications enabled by the CTS structure.

¹See [17] or https://github.com/cite-architecture/cts_spec/blob/master/md/specification.md.

2 Canonical Text Services

Canonical Text Service is a protocol developed in the Homer Multitext Project ([17]) and "defines interaction between a client and server providing identification of texts and retrieval of canonically cited passages of texts" by using CTS URNs as persistent and location-independent resource identifiers. Generally, CTS describes a web service that is able to serve text passages based on a URN reference system.

The specifications do not limit the communication to HTTP. Yet, every known implementation uses HTTP as its information transport protocol. Each HTTP request has to include a parameter *request* which specifies, what function of CTS is used. Possible values are *GetPassage*, *GetPassage-Plus*, *GetValidReff*, *GetCapabilities*, *GetFirstUrn*, *GetLabel* and *GetPrevNextUrn*. Parameters for individual function calls are added as parameters to the HTTP request. Whether or not GET or POST is used to communicate parameters is not restricted by the specifications. Yet, every known implementation uses GET, which also seems reasonable considering the length of the expected values and the use case of persistent hyperlink references in digital documents. Additional parameters that are not part of the protocol specifications may be added to the request, as is explicitly done in the specifications with the use of the example

```
http://myhost/mycts?configuration=default
&request=GetCapabilities
```

Such parameters may be used to provide a configuration for the service.

2.1 CTS URNs

A URN of the CTS protocol is a persistent identifier that specifies exactly one text passage. Every CTS URN must start with *urn:cts:* followed by the three components *{NAMESPACE}*, *{WORK}* and *{PASSAGE}*. The components are divided from each other with a colon, resulting in the syntax schema *urn:cts:{NAMESPACE}:{WORK}:{PASSAGE}*.

{NAMESPACE} specifies the namespace of the text collection and can be used to separate one text data set - or text corpus - from another.

2.1.1 The *{WORK}* component

{WORK} specifies the document and is further divided into four parts that are based on the Functional Requirements for Bibliographic Records ([10]). The four parts are *TEXTGROUP*, *WORK*, *VERSION* and *EXEMPLAR* and must be specified in the given order. Note that *{WORK}* and *WORK* describe two different components with a very similar name. Each value must be a valid entity in its surrounding context. This means that the given *VERSION* must exist for the specified *WORK* and the given *WORK* must exist in the given *TEXTGROUP* and so on. The components of *{WORK}* are separated from each other with a dot. A complete CTS URN for a *{WORK}* based on a English bible translation in the Parallel Bible Corpus may be

```
urn:cts:pbcbible.parallel.eng.kingjames:
```

TEXTGROUP is the only mandatory component. Any other component may be missing if it is not followed by another component. For instance,

```
urn:cts:pbcbible.parallel.eng:
```

and

```
urn:cts:pbcbible.parallel:
```

are still valid URNs while

```
urn:cts:pbcbible.parallel.kingjames:
```

is not because it omits the ".eng" part. The missing parts can be completed arbitrarily by the implementation. If the URN

```
urn:cts:pbcbible.parallel:
```

is used as a reference, the response may be based on

```
urn:cts:pbcbible.parallel.eng.kingjames:
```

or

```
urn:cts:pbcbible.parallel.deu.luther1545:
```

2.1.2 The *{PASSAGE}* component

{PASSAGE} specifies the text passage in the document and consists of an unspecified number of elements separated by a dot. The components of *{PASSAGE}* relate to the structural elements of the document like chapter, paragraph or sentence. Figure 1 illustrates such a hierarchical document structure based on a bible translation.

Figure 1. Hierarchical Document Structure

The type and label of these elements is not limited by the specifications and may vary between different documents based on their genre and other properties. While stanza and verse are intuitive structural elements for a common poem, this is not the case for a scientific paper, which would probably be better structured in chapters, sections, paragraphs and sentences. A CTS URN that includes a *{PASSAGE}* element may look like the following example.

```
urn:cts:pbcbible.parallel.eng.kingjames
:35.1.10
```

2.1.3 Usage of CTS URNs

Static CTS URNs allow to reference any of the structural elements of the document. For example, the CTS URN

```
urn:cts:pbcbible:parallel.eng.kingjames:35.1.10
```

refers to the sentence 10 of chapter 1 of book 35 of the English King James bible translation as illustrated in figure 2.

Figure 2. Hierarchical Document Structure 35.1.10

The values of the references are not limited to numbers - the CTS URN

```
urn:cts:pbcbible:parallel.eng.kingjames:genesis.3rdDay.5thsentence
```

would also be a valid reference. This especially means that numbers do not necessarily reflect the document order. The passage 2.2.3 may be followed by the passages 2.2.1 or 1.1.1.

Similar to the *{WORK}* component, the *{PASSAGE}* must not be fully specified and parts may be missing if they are not followed by other components. The text passage is always resolved as far as it is specified. For the given example, 35.1 would refer to the complete chapter 1 of book 35 and 35 would refer to the complete book 35. The CTS URN

```
urn:cts:pbcbible:parallel.eng.kingjames:35
```

refers to the complete 35st book of this bible translation as illustrated in figure 3.

Figure 3. Hierarchical Document Structure 35

The number of components in the *{PASSAGE}* is often referred to as citation depth. This value is not limited by the specifications. This means that - contrary to the *{WORK}* component - the number of elements in the *{PASSAGE}* component is not limited, which is an important detail to consider when working with CTS URNs.

In addition to static references for fixed structural elements, CTS allows to refer to text passages more dynamically using sub passage notation and spanning CTS URNs. Sub passage notation allows to refer to specific parts of a text passage, like a word or even a character. Sub passages are marked with "@". The number of occurrence of the reference point can be specified in brackets "[]" or omitted if it is the first occurrence. The CTS URNs

```
urn:cts:pbcbible:parallel.eng.kingjames:1.3.5@be
```

```
urn:cts:pbcbible:parallel.eng.kingjames:1.3.5@be[1]
```

both refer to the first occurrence of "be" in the text passage. If sub passage notation refers to a sub passage that does not exist, then this CTS URN is not valid.

The *{PASSAGE}* component of a spanning URN consists of two passage references that are separated by "-" like in the CTS URN

```
urn:cts:pbcbible:parallel.eng.kingjames:35.2-35.3
```

which references the text passage illustrated in 4.

Figure 4. Hierarchical Document Structure 35.2-35.3

These CTS URNs specify the text passage that spans from the left structural element to the right including the text of each of them. Both references must not be fully specified and may include sub passage notation as in the CTS URN

```
urn:cts:pbcbible:parallel.eng.kingjames:35.2@upon-35.3@my[3]
```

that corresponds to the text passage roughly illustrated in figure 5.

Figure 5. Hierarchical Document Structure 35.2@upon-35.3@my[3]

Due to the support for sub passage notation and spanning CTS URNs it is ensured that every possible text passage of

every document can be referenced with CTS URNs. The worst case scenario would be a document that is not structured in any way and therefore consists of only one structural element. Still, using spanning CTS URNs and sub passage notation, every text passage in this document can be referred to by referring to the span between two words in this one structural text element.

2.2 Functions

The other important part of the CTS protocol is a set of functions that the service must provide. As the development of CTS continues, additional functions may be added. This section describes the functions as they are defined in CTS 5 rc.1.

Generally, the function name and parameters are requested as parameters for the service as expected by the communication protocol that is used. This implementation relies on HTTP communication and uses GET parameters. The benefits of this approach can be summarized as widespread technical support and persistent and share-able bookmarks in the form of commonly known hyperlink URLs. For instance, a CTS request for this implementation that requests the text passage for a given CTS URN can look like the following HTTP request.

```
http://myhost/mycts?request=GetPassage&urn=
urn:cts:pbcbible.parallel.arb.norm:1
```

The result of a function call is a XML document similar to the following schema.

```
<{FUNCTIONNAME}>
  <request>
    {PARAMETERS}
  </request>
  <reply>
    {RESPONSE}
  </reply>
</{FUNCTIONNAME}>
```

Each of the functions is compatible with any kind of CTS URN including dynamic CTS URNs that span two static CTS URNs or CTS URNs that use sub passage notation.

2.2.1 GetCapabilities

GetCapabilities returns the text inventory of an instance of CTS with all the CTS URNs of works and editions as well as meta information for each entry. The extend or content of the meta information is not specified in CTS. The text inventory is divided into several text groups that contain several editions with correlate to one (physical) document.

2.2.2 GetValidReff(urn,level)

GetValidReff returns all the CTS URNs that belong to the given CTS URN. The required parameter *level* specifies the maximum citation depth of the CTS URNs in the response. The response is a list of CTS URNs that may be empty if no suitable CTS URNs are found for the provided citation depth.

2.2.3 GetLabel(urn)

GetLabel returns an informal description of the given CTS URN. The way that the URN is described is not specified and can differ between different implementations of the protocol. This implementation translates the URN to common language using the information that is available in the system. For example the URN

```
urn:cts:pbcbible.parallel.eng.kingjames:1
.2-1.5.6
```

is translated to *King James Version of the Christian Bible from book "1", chapter "2" to book "1", chapter "5", book "6"*.

2.2.4 GetFirstUrn(urn)

The definition of this function is ambiguous. According to the CTS protocol specifications, GetFirstUrn "identifies, at the same level of the citation hierarchy as the urn parameter, the first citation node in a text.". This can either mean that it identifies the first child URN of the given CTS URN or the first child URN that actively contains text. If the first child URN refers to a chapter that is further divided into structural elements - like sentences - then this definition may be meant in a way that GetFirstUrn identifies the first sentence because it contains text content. But since the chapter also contains text in the form of the sum of the texts of the sentences, it could also identify the chapter. Returning the first child URN of a given CTS URN whether or not it contains text was the approach that seemed more intuitive and therefore was chosen in this implementation.

2.2.5 GetPrevNextUrn(urn)

GetPrevNextUrn returns the previous and next URN in document order for the given urn. This function makes sure that the URNs are in sequential order but in most practical cases it would probably be more efficient to use GetValidReff with the parent CTS URN instead of iterating through the sequence of CTS URNs using singular GetPrevNextUrn requests.

2.2.6 GetPassage(urn,[context])

GetPassage returns the text passage that belongs to a given CTS URN. The parameter *context* is optionally specifying, how many text units should be added to the passage as contextual information. Since text is not limited to valid XML but this response has to be formatted as valid XML, the text content is XML escaped per default in this implementation. If this is not wished, the escaping process can also be prevented, but then the XML validity of the response can not be guaranteed.

2.2.7 GetPassagePlus(urn,[context])

GetPassagePlus returns the combined results from the other functions except GetCapabilities. The rules for the

other functions also apply except for one difference: *GetValidReff* requires the parameter *level* to be set that is not specified for *GetCapabilities*, which is a potential oversight in the specifications since they state that the "contents of the *validreft* element must be the same as the corresponding *GetValidReff* request.". The response is the sequential listing of the various results.

3 Canonical Text Service in (Digital) Humanities

Since the CTS protocol was developed in the Digital Humanities, it can be assumed that it is of significant importance in this research area. Yet to position this work and also to potentially introduce the protocol to the audience of the Not-Yet-Digital Humanities, it is important to describe the requirements and potential benefits that can be assumed to be expected when researchers from the humanities are confronted with Canonical Text Services. It is important to note that this work is written from a technical point of view and the benefits and requirements listed in this section cannot be expected to be comprehensive. There may be important aspects that are not covered. Yet, this requirement analysis hopefully provides a competent and understandable overview and helps researchers in the humanities to relate this work to their research field while illustrating the position of this work for researchers in Digital Humanities that are already familiar with the CTS protocol.

3.1 Requirements on CTS in the (Digital) Humanities

The protocol was developed in the Digital Humanities and therefore can be expected to reflect humanistic requirements on such a reference system. This means that requirements like persistence, citability and technology independence can be deduced directly on basis of the CTS specifications and its surrounding discussions. Additional requirements - like support for licensing and good usability - are the result from direct feedback from researchers in the Humanities.

3.1.1 Persistence and Citability

Persistence is explicitly required by the CTS URN specification, as it defines CTS URNs as "persistent, technology-independent identifiers for texts and passages of texts". This means that every CTS URN must always refer to the exact same text passage. This also implies that each CTS URN may only refer to one or zero text passages. The purpose of this requirement is to guarantee citability. To be able to cite a text passage, it is obviously required that referenced text content is not allowed to change.

Since the other two parameters - *context* and *level* - are functional parameters that specify, how certain functions must behave, persistent CTS URNs guarantee a persistent Canonical Text Service. The only cases in which CTS URNs would not fulfil the persistence requirement are changing

text contents or duplicate URNs. Changed text contents can only occur if the server administrator changes text manually, which would be irresponsible behaviour. Duplicate CTS URNs can only occur if they use the same namespace. This can only happen if they belong to the same text corpus or if an already established namespace is used by another text corpus. Preventing both cases is the responsibility of the text editors and the reuse of existing namespaces can be technically prevented with the Namespace Resolver described in section 5.5.3.

3.1.2 Granularity

The granularity of a reference system describes the level of detail that its references provide. For instance in the context of text data, *document level* is a commonly used granularity. For text passages it is not only required that references can be resolved on the smallest granularity but also on every higher level. For example, being able to reference individual words in a text would indicate a very fine-granular reference system. Yet CTS must also support references for every structural element that leads to this word like the enclosing sentence, chapter or book.

Fixed structural elements can be accessed using static URNs. Using spanning URNs and sub passage notation, every text passage can be referenced on character level, which is a reasonable smallest unit for text reference.

Granularity can also be applied to the documents of a text corpus. Several editions or translations can be collected as one work and several works can be collected in one text group. An example would be a text group consisting of the works of William Shakespeare. The work level could refer to Shakespeare's Hamlet and be further divided into several translations and editions. One digital representation of a physical book of this work can then be referred to as a certain translation or edition. The work part of a CTS URN must always be specified at least to the text group level. In the case of missing elements in the work part the implementation is free to choose any fitting version of the work. This means that document granularity is resolved as *choose any* which is different from text passage granularity that is resolved as an composition.

3.1.3 Licensing

Licensing is not covered by the Canonical Text Service protocol. Yet personal feedback strongly suggested that detailed license handling for every possible request must be included because each passage must be considered as a citation of the licensed texts. Licenses like Creative Commons *CC-BY* ([4]) require that every citation of the document must include the license, a reference to the publisher and a statement if the text has been modified. This means that to be able to serve texts that are licensed as *CC-BY* or similar, it must be technically possible to serve this information along with every CTS response.

The developed solution is unique to this implementation and includes a license text and reference that the editor for-

mulated in the source file. Since serving the text via CTS is technically a modification, a reference to the CTS instance and the CTS URN are added to the source element. The information is accumulated if more than one source or license is provided. This may result in relative complicated long entries as the source that is shown in figure 6.

```
<reply>
  <urn>urn:cts:cta:leutwein.einladungsschrift.de.norm:1</urn>
  - <passage>
    Einladungs-Schrift bei dem feierlichen Rede-Akt welcher den 19ten April in allhiesigem Gymnasio von drei Zöglingen
    welche die Akademien beziehen wollen soll gehalten werden von Lorenz Friedrich Leutwein Rektor Gymnasii,
    Professor, und Contubernii Ober-Inspektor Bibliothekar.
  - <passage>
  - <license>
    Distributed under the Creative Commons Attribution-NonCommercial 3.0 Unported (German) License.
  - <license>
  - <source>
    http://www.deutschestextarchiv.de/leutwein_einladungsschrift_1797 http://www.deutschestextarchiv.de
    /book/download_xml/leutwein_einladungsschrift_1797 http://www.deutschestextarchiv.de/book/download_html
    /leutwein_einladungsschrift_1797 http://www.deutschestextarchiv.de/book/download_text
    /leutwein_einladungsschrift_1797 leutwein_einladungsschrift_1797 30397 urn:nbn:de:kobv:b4-30397-7
    http://www.deutschestextarchiv.de/leutwein_einladungsschrift_1797 http://www.deutschestextarchiv.de
    /book/download_xml/leutwein_einladungsschrift_1797 http://www.deutschestextarchiv.de/book/download_html
    /leutwein_einladungsschrift_1797 http://www.deutschestextarchiv.de/book/download_text
    /leutwein_einladungsschrift_1797 leutwein_einladungsschrift_1797 30397 urn:nbn:de:kobv:b4-30397-7 retrieved via
    Canonical Text Service www.urncts.de/cta/cts with CTS URN urn:cts:cta:leutwein.einladungsschrift.de.norm:1
  - <source>
</reply>
```

Figure 6. License example from CTS based on data from the Deutsche Textarchiv

Accumulation is considered as a better solution than a more or less arbitrary choosing algorithm because it gives the editors the control about the license text content.

3.1.4 Unrestricted Access

The specifications make no mention of any kind of user management or access restriction. This and the required citability suggest that the access to the web service must be unrestricted. CTS URNs as citable references in digital documents - similar to URLs that are added to documents as common practice for years - can only be of benefit if the corresponding resource can be accessed and restricting this access limits the benefit of the references.

If required, restrictions can still be applied by the server administrator who is responsible for a specific CTS instance. Making sure that the access policy is not contradicting the licenses of the texts is also the responsibility of the corresponding document editors.

3.1.5 Technology Independence

The CTS URN specifications define CTS URNs as "persistent, technology-independent identifiers". This means that a Canonical Text Service has to be technology independent by definition. Any implementation of CTS must make sure that it does not restrict the rules defined in the protocol and it must always be possible to replace one implementation of CTS with another without any visible changes in the functionalities of the web service. This especially means that a CTS implementation should not be limited to any text format like for instance TEI/XML.

While this implementation uses TEI/XML as its import format, this is not a technical requirement. Import work flows can also be created for any other text format. Any kind of CTS URN that is mentioned in the specifications is supported and the functions work as they are described. The output is formatted according to the specifications and

served via HTTP communication. Like any other computer program, the implementation has certain technical requirements but none of them interfere with the functionalities that are described in the CTS specifications.

Various technical features are unique to this implementation and the basis for some of the tools and applications that are described in chapter 5. These features have to be excluded from this requirement because they are not part of the specifications and must be considered as external services.

3.1.6 Usability

Usability describes the difficulty that a user has to expect when using a certain technical solution and is a very relative and subjective requirement. Since the target audience for CTS partly consists of researchers that are not focused on and may not be well trained in Computer Science, it can be assumed that this difficulty and the additional work load that is required to set up a Canonical Text Service should be as minimal as possible. This is especially important because the specifications themselves and the concept of a web service already provide a relatively steep learning curve.

One of the tools described in chapter 5 is a graphical management tool for instances of CTS on an existing server that was developed to meet this requirement. Additionally it is made sure that this implementation only depends on requirements that are usually met by any common server - namely up to date versions of JAVA and MySQL.

3.1.7 Language Independence

Language Independence is considered as a very important requirement for a Canonical Text Service, especially if it is supposed to work as an unifying infrastructure element as described in [22]. A text communication protocol that limits the possible text content for technical reasons - that may be unfamiliar for researchers in the Humanities - is obviously not helping much in a research field that includes multi lingual text corpora like the Parallel Bible Corpus ([11]) with its 903 included languages.

Per default, this implementation uses UTF-8 ([23]) as its character set, a system that uses 1 to 4 8-bit sequences to encode characters. UTF-8 is commonly used in digital documents and is per default supported in JAVA. It provides support for almost all commonly used alphabets including Latin, Cyrillic, Coptic, Arabic, Armenian, Hebrew, Greek, Syriac, Japanese, Chinese, Korean and even mathematical symbols and emojis. The character encoding can be configured to use any encoding that is supported by MySQL. This means that - if necessary - CTS instances can also support UTF-16 or UTF-32.

It is also possible to limit the character encoding for a given CTS instance to a smaller alphabet like Latin to potentially improve the data efficiency. But basic tests did not show any noticeable benefit in doing this, especially with respect to the fact that encoding related problems can create unnecessary difficulties.

Aside from character encoding, another language related issue is important: the direction of the text content. Latin based languages are written and read from left to right (LTR). Arabic based languages are written and read from right to left (RTL). This is especially problematic because CTS responses are served as documents that use a Latin based - LTR - XML markup. When serving Arabic text passages, the resulting XML document is a mix of LTR and RTL text that may include different text directions in the text passage or even in the used CTS URNs². This mix of directions makes it very hard to process such character strings as it is for example illustrated in varying interpretations of the same text passage by different tools in figures 7, 8 and 9. Figure 7 shows a screenshot of a text passage in Microsoft Word that mixes Arabic and Latin alphabet.

```
<edition urn="urn:cts:alimonthly:digital.muqtabas_Edited.TEIP5:">
<title>سبب قجلا تلجم Majallat al-Muqtabas TEI edition</title>
<author>Muammad Kurd Al</author>
```

Figure 7. LTR/RTL mixed Text Passage in Microsoft Word

When this text passage is copied into another editor, like Notepad++, the results looks different. It looks like the Arabic text content is inverted compared to the passages from figure 7.

```
<edition urn="urn:cts:alimonthly:digital.muqtabas_Edited.TEIP5:">
<title>مجلة المفتيس Majallat al-Muqtabas TEI edition</title>
<author>Muammad Kurd Al</author>
```

Figure 8. LTR/RTL mixed Text Passage in Notepad++ v6.8.8

When exported to PDF format and opened in a PDF Viewer, another version of this passage occurs.

```
<edition urn="urn:cts:alimonthly:digital.muqtabas_Edited.TEIP5:">
<title>la tallajaM مجلة الامة ت بس Muqtabas TEI edition</title>
<author>Muammad Kurd Al</author>
```

Figure 9. LTR/RTL mixed Text Passage in PDFXChangeViewer 2.5 Build 313.0

Because of language barriers, this can not be investigated in detail but it seems that there are some problems here. The PDF document illustrated in figure 9 definitely uses the wrong direction for part of the Latin based text and two versions of the Arabic text content seem to use different directions.

This problem has to be solved outside of this work and probably requires a lot of further and specialized discussion. Tests showed, that RTL and LTR are supported by this implementation for static and dynamic CTS URNs and text content. But since different tools seem to interpret the text content differently and the human interpretation depends on the result that is shown by these tools, it is advised to be careful especially with bi-directional text content that is served by CTS or any similar system.

3.2 Benefits of CTS support in the (Digital) Humanities

To argue for an integration of Canonical Text Services in the Humanities it is important to show that this integration would provide significant benefits for the work that is done in this research area. Some of these benefits - like the ability to reference digital text uniformly and with flexible granularity - can be directly implied by the purpose of CTS. Other benefits - like its use as a text archive and open publication platform - are not explicitly related to the specifications but can still be implied from current day trends, practises and related research projects.

3.2.1 Archival

Archival of (text) documents is an important aspect of Digitization in general. On the one hand (Group A) Projects like Das Digitale Archiv NRW([18]), CLARIN([7]) and the Internet Archive³ are created to provide technical infrastructures for this purpose and solve various problems associated with it including link rot, backup handling, access handling, versioning and many more. On the other hand (Group B), document digitization projects like Perseus, Das Deutsche Textarchiv, The Parallel Bible Corpus and Croatiae Auctores Latini use existing technologies to provide project specific solutions for their project specific data sets including the use of publicly available tools like sourcecode repositories⁴ as well as hand crafted solutions⁵. Three problems can be identified because of this setup:

- There exists a gap between the goal for generalized and widely applicable solutions that can be assumed for group A and the project specific solutions that are not uncommon for group B.
- Once a digitization project is finished and the promised data set is created, it becomes hard to argue for the continuation of the project and therefore for further funding. If the data set is finalized, the further required work is focused on technical issues like website maintenance and archival, which is probably not a suitable task for the same researchers that were required for text editing. Without funding, it becomes hard to keep the project running and maintained. For example the project *Briefe und Texte aus dem intellektuellen Berlin um 1800* provides a contact email address that is no longer working⁶ and the API for the TED Talk Transcripts was closed in 2016.
- Since archival also introduces technical issues, it can be assumed that a lot of problems are already solved by group A that are not yet solved or even known by group B. Example issues include link rot, storage and backup techniques, access management, versioning and format

³<https://archive.org/>

⁴Perseus

⁵Das Deutsche Textarchiv, Parallel Bible Corpus

⁶<http://www.berliner-intellektuelle.eu/imprint>

²Sub passage notation

conversion. While such issues are part of the problems that group A wants to solve for a possibly wide audience, project specific solutions by group B may not even be aware that such issues exist. This potentially creates solutions that include problems that could have been prevented like data loss because of a missing technical backup work flow.

The Canonical Text Service protocol can serve as an uniform communication interface between group A and group B and also provide a competent archival solution in itself. Because CTS URNs are by definition technology independent - and therefore implementation independent - implementation specific problems like link rot and the required reliance on a specific website or storage technique are not relevant. A CTS URN always refers to the same text passage and is independent from the server. If a project specific server can not be maintained any longer, then another server can serve the same data using the same CTS URNs. The only thing that needs to be adapted is the address of the new server, which corresponds to one entry in a registry.

Using the principle of CTS Cloning (See [20]), this implementation of CTS can also be used as a decentralized backup method for text data. It is possible to copy the content from one CTS instance into another and this way create a backup or include a given data set in an already existing infrastructure. This way it is possible to create a data environment that combines decentralized and centralized data storage into an infrastructure that provides reliable and supervised central archives as well as flexible project specific backup solutions for projects that may not (yet) be authorized to be included in the established central archives.

3.2.2 Interoperability

Interoperability in the context of software generally describes the amount to which it is possible to interchange work flows and data sets. In a completely interoperable environment, it is possible to combine any work flow with any data set. For practical reasons⁷ this can rarely be achieved. Especially research areas with limited technical focus like the Humanities tend to produce project specific solutions for their research question. This heterogeneity of solutions is a disadvantage because it results in a lot of redundant work flows. While not all research work can or should be generalized and especially research projects in the Humanities tend to require a lot of domain specific knowledge and work flows - that may even contradict each other - it is obviously an advantage to generalize basic tasks like token statistics and search engine support and also provide generic interfaces for more complex tasks like citation analysis and topic models. These generic solutions do not have to provide perfect solutions for any given combination of data set and work flow but merely default starting points and basic results that a researcher can tweak for the given context.

To achieve interoperability between work flows and data sets it is required to create a common communicative ground. This can either be achieved by increasing the number of supported data formats or by providing an uniform interface. The Canonical Text Service protocol can serve as such a uniform interface and provides online access to the data as well as generic request functionalities.

Additionally, since CTS URNs are technology independent by definition, they provide a service independent way of referencing text passages. An established persistent CTS URN will always reference the same text passage, even when it is used to request data from another server. This especially means that results from different research projects can be combined and can complement each other. This is especially relevant for research projects work flows that require a lot of project specific optimisation and manual work like for instance citation analysis. With an interoperable reference system for text passages, it is possible to include the results of existing citation analysis work flows instead of having to start from scratch. It may also be reasonable to combine the results from various approaches if they produced different citations.

3.2.3 Text References

As the main purpose of CTS URNs, this benefit is obviously important. CTS URNs allow to reference any possible text passage of a document as a persistent and shareable resource. As long as the data access is not restricted by the server administrator, CTS also provides open online access to the data. This makes it possible to easily share specific text passages that are relevant for a discussion. When used in manual editing work flows, CTS URNs can also provide a uncomplicated way to distribute text fragments over various researchers and editors.

A practical academic benefit is that these references can be used to cite text passages in digitally produced documents similar to how URLs are used for this purpose. These references can be included in the form of a URL⁸ or CTS URN⁹. When used in documents that are only available digitally, it is also possible to hide the relatively complex URLs for CTS requests behind hyperlinks as it is commonly done in the world wide web.

Using the text alignment techniques described in section 5.2, it is even possible to retrieve a shared text passage in a different document and this way retrieve a translation of the text passage. Given that the document is structured uniformly in multiple translations this can prevent problems that result from language barriers.

3.2.4 Publication

A Canonical Text Service provides public access to its documents and can be set up on a common server. This means

⁷data format specific features, data types, expectations with regard to the data, research contexts,...

⁸<http://cts.informatik.uni-leipzig.de/pbc/cts/?request=GetPassage&urn=urn:cts:pb:bible.parallel.eng.kingjames:1.3>

⁹urn:cts:pb:bible.parallel.eng.kingjames:1.3

that it is possible to independently publish texts as public resources using a private server. Technically this can also be done by simply storing a text document on a server but using CTS provides the benefit that generic tools like the ones described later in this document can be used to serve and work with the published documents. This way publishers can simultaneously provide public access to the documents and valuable input for research data. While this is not suitable for documents that are still sold commercially, it might be an attractive way to generate public interest and benefit from non-commercial documents and - for instance - outdated newspapers.

3.3 Canonical Text Service as a Communication bridge between Digital Humanities and Computer Science

In text related Digital Humanities, Canonical Text Services are a suitable candidate for an interface that bridges this gap between referencing and accessing text because it is a communication protocol that reflects the requirements that humanists have defined for such a system. With a working implementation of the protocol, tool developers can rely on a normalized interface that can be referenced when developing tools, and that is agreed on by at least part of the humanities research community. This results in less work that is required to make text data sets compliant for tools and frameworks.

Additionally, since the protocol was developed in the Digital Humanities, it can be assumed that researchers in this area want to use this system, and preparing the data sets is a research task in this area. CTS compliant data is generated in the Digital Humanities whether or not it is used in Computer Science. This means that supporting this type of data source creates an environment in which the data does not have to be prepared to be compliant with implemented systems but instead the implemented systems use a data source that grows independently.

4 Available Datasets

The following data sets are currently available as public CTS instances. Including more data sets is one of the main tasks that has to be done as future work and interested researchers are always welcome to suggest or include new data sets.

4.1 Deutsches Textarchiv & Textgrid

The Deutsche Textarchiv ([5]) and Textgrid([13]) provide German contemporary literature. Textgrid spans a time period from 1200 to 1998 and also covers translations of non German works. It contains 62281 documents from 689 authors. The document size ranges from small letters to large works of literature like the translation of Tolstoi's *War and Peace*. The Deutsche Textarchiv focuses on German authors and includes 2730 documents from 1205 authors. The doc-

uments are available in three normalization stages and were written in the time period from 1465 to 1927.

4.2 Parallel Bible Corpus

The Parallel Bible Corpus ([11]) contains 837 parallel bible translations in 817 languages. Because of licensing requirements, only 20 translations are available as public resources.

4.3 TED Talk Transcripts

TED Talks¹⁰ are a collection of video lectures that cover various topics. The videos have been transcribed in 105 languages by the listening community to provide subtitles. The CTS instance contains 52987 transcripts that cover 1938 of the talks. Since the access to this resource is no longer maintained by TED and there does not seem to exist an alternative source for the data, this data set is a great example for the archival use case of CTS.

4.4 Perseus

Perseus ([16]) is a manually edited collection of Greek and Latin texts that is closely related to the development of the CTS protocol and therefore is edited in a way to specifically suit its requirements. It is publicly available as a regularly updated GitHub repository¹¹. The CTS instance provides 2126 documents.

4.5 Others

In addition, various smaller and more focused text collections are available as public CTS instances including contemporary letters from around 1800 in *Briefe und Texte aus dem intellektuellen Berlin um 1800* ([1]), Croatiae Auctores Latini ([9]), German political speeches from 1984 to 2011 in the *German Political Speeches Corpus* ([2]), the Arabic monthly journal *Ali's monthly journal al-Muqtabas* ([6]) and a collection of the German and English versions of the works of William Shakespeare.

5 Applications

This chapter describes the software tools and applications that were developed in the context of this work. These include a reader as an easier and more comfortable way to retrieve and work with CTS data, text mining tools that include commonly established analytical techniques as well as unique and new algorithms that build on parts of this work and management tools that are implemented to make it as easy as possible to create and manage instances of Canonical Text Services on a given server.

¹⁰<https://www.ted.com/>

¹¹<https://github.com/PerseusDL>

The tools and applications in this chapter in part rely on the advanced functionalities that are provided by this implementation because they could not be implemented with only the methods that the CTS specifications provide. This means that, while it may be possible that certain features are working across various implementations, general compatibility should not be expected.

To be more appealing for users, most tools provide a helpful Graphical User Interface (GUI).

5.1 Canonical Text Reader and Citation Exporter([15])

Retrieving the data from and specifying the reference points in a Canonical Text Service using only the functionalities it provides can be considered as a relatively complicated thing to do. Generally, it is required to work with potentially complex URLs that include relatively complex CTS URNs as parameters. These URLs are the only way to request data and have to be created manually to do so. Furthermore there does not exist a trivial way to create a CTS URN suitable for a given text passage because CTS is only designed to serve text based on a given CTS URN and not the other way. To create a suitable CTS URN, it is required to retrieve an estimated context of static URNs¹² surrounding a text passage and then guess the exact URN or span of URNs.

CTRaCE was developed to provide a more user friendly way to work with the data in CTS related research. As such it should not be considered as software for end users but a visualisation tool for CTS. Figure 10 shows the GUI.

Figure 10. CTRaCE

The top hierarchy is used to select a document based on the {WORK} component. On the left side next to the text area the structure of the document is illustrated based on the {PASSAGE} components of the document. The text area can either include the raw text including the XML meta data markup or a TEI/XML based styled view of the text as illustrated in figures 11 and 12.

Figure 11. CTRaCE Styled View

Figure 12. CTRaCE XML View

Text in the text area can be selected using a computer mouse or any other common selection method and the button "Export Citation" provides CTS URNs for the selected text passages and the corresponding HTTP links to CTRaCE itself and the CTS server.

CTRaCE is included in and deployed by the administration tools described in this chapter.

5.2 Using CTS URNs for real time text alignment

By extending the request functionalities of the MySQL based implementation a text alignment method was implemented that uses structural information implicitly contained in CTS URNs to align text passages in real time. This method also provides a solution to one of the general systemic problems of current text alignment methods - the accumulation of error probability that happens as more and more text parts of a document are aligned.

5.2.1 Problem Description

Text Alignment or Text Passage Alignment is the task of finding comparable text parts in different documents. The meaning of the word comparable depends on the goal that is set. If the goal is to find cases of Text Reuse - e.g. Plagiarism - then the similarity of two text passages is important but it is less important, where in the document these text passages are

¹²GetValidReff or GetPrevNextUrn

located. If the goal is to find parallel text parts, then the location of the text passages is very important, but the similarities between the text passages can be ignored.

There are multiple approaches to align text passages for Text Reuse like n-gram based algorithms as for instance described in [14]. For the second goal, these methods cannot generally be applied for multilingual data sets, because it cannot be assumed that the languages in which the documents are written in are known or that they use the same character set. In the English Darby translation of the bible in the Parallel Bible Corpus, the first sentence from book 50 is

Paul and Timotheus , bondmen of Jesus Christ ,
to all the saints in Christ Jesus who are
in Philippi , with [the] overseers and
ministers ;

The text passage in the corresponding document for the French David Martin translation is

Paul et Timothee , Serviteurs de Jesus-Christ ,
a tous les Saints en Jesus-Christ qui sont
a Philippi , avec les Eveques et les
Diacres .

N-gram based algorithms would probably start to struggle here.

While this alignment can be found using similarities in the Named Entities used in this sentence, it may happen that other sentences share this set of Named Entities too. Additionally, since they require pre defined training data sets, comparing Named Entities becomes tricky when working with unfamiliar languages that may include varying character sets.

Another problem of text alignment is that the parallel alignment is getting increasingly unreliably as the document continues, because the probability of an error in the alignment accumulates with each new aligned pair.

Document structure is part of the information that is added to digitized works in the manual editing process and it implicitly encodes the parallel text alignment. Based on this structural information - that is also used to generate the static CTS URNs - it is possible to use CTS to find parallel text parts without knowledge about the languages that the documents are written in and without accumulating the error probability. By extending the functionalities of CTS, the implicit alignment information is made available without any need for separate pre calculation.

5.2.2 Properties of CTS URNs

CTS URNs have several unique properties that are important for the text alignment process.

Property (1) is that if the *{WORK}* part of an CTS URN is exchanged and the *{PASSAGE}* part is kept, CTS returns the same text passage from another document. The URN

```
urn:cts:dta:krueger.weltweisheit1746.de.transcript:5
```

refers to sentence 5 of the transcript exemplar of Kruegers "Weltweisheit". The URN

```
urn:cts:dta:krueger.weltweisheit1746.de.norm:5
```

specifies the same text passage in the normalized exemplar. The hypothetical URN

```
urn:cts:dta:krueger.weltweisheit1746.en.norm:5
```

would return the same text passage in the English version of the document.

Property (2) is that typically the last two elements specify the language and the exemplar or edition in this language. This means that if the URN up to the second last element is specified, the resulting document options are a set of documents that are different exemplars of the document in one language. This, of course, only works if the data is prepared in such a way that the second last element of the *{WORK}* of the URN is the language tag. This must technically not be the case.

Property (3) is the hierarchical system of persistent IDs that is generated in the *{PASSAGE}* part. The text part 2.3.5 is part of 2.3 but 2.3 is not part of 2.3.5. This means that for each text part, the sequence of text parts that are part of it, always starts new. The reference to the first sentence of the fifth chapter refers to the first sentence in the fifth chapter for any document that is structured accordingly.

5.2.3 Text Alignment

Using these properties, two methods for text alignment were implemented.

Parallel Alignment requires one CTS URN A, which is used to specify the text passage and a set S of URNs that specifies, which documents will be aligned against this text passage. Each of the URNs in S is combined with the *{PASSAGE}* of A and the text for this URN is added to the result set. The assumption is that because of property (1), all the text passages that are specified by the generated CTS URNs refer to the corresponding parallel text passages in another document.

The following example illustrates the steps that are done. Using the *{PASSAGE}* 43.20.28 makes sure that a text passage is chosen from somewhere in the middle of the document. The URN

```
urn:cts:pbcbible.parallel.eng.darby:43.20.28
```

refers to the text passage

Thomas answered and said to him, My Lord and my God.

The following set of documents will be aligned against this passage:

```
urn:cts:pbcbible.parallel.fra.kingjames:
urn:cts:pbcbible.parallel.deu.luther1912:
urn:cts:pbcbible.parallel.mya.1835:
urn:cts:pbcbible.parallel.rus.synodal:
urn:cts:pbcbible.parallel.ceb.bugna:
urn:cts:pbcbible.parallel.ukr.2009:
```

The text passages for the following URNs will be requested:

```
urn:cts:pbcbible.parallel.fra.kingjames:43.20.28
urn:cts:pbcbible.parallel.deu.luther1912:43.20.28
```

urn:cts:pbcbible:parallel.mya.1835:43.20.28
 urn:cts:pbcbible:parallel.rus.synodal:43.20.28
 urn:cts:pbcbible:parallel.ceb.bugna:43.20.28
 urn:cts:pbcbible:parallel.ukr.2009:43.20.28

The result is the alignment shown in figure 13.

Thomas répondit et lui dit : Mon Seigneur et mon Dieu !
 Thomas antwortete und sprach zu ihm : Mein HERR und mein Gott !
 သောမကလည်း ၊ အကျွန်ုပ် ၏ အရှင် ၊ အကျွန်ုပ် ၏ ဘုရားသခင်ပါလောကားဟု ပြန်၍ လျှောက် ၏။
 Φ ο μ α. σκαζαλ Εμυ ε ν α ν τ ω σ τ : Η ο σ π ο δ υ ὡ μ ο ι κ α ι Η ο σ ο μ ο ι κ α ι
 Ug si Tomas miingon kaniya , “ Ginoo ko ug Dios ko ! ”
 А Хома відповів і сказав йому : Господь мій і Бог мій !

Figure 13. Parallel Text Alignment Example

Checking the results with the Translation API of Google shows that each of these text passages translates to the originally specified passage.

Figure 14 shows a tool that creates a matrix that aligns the individual text parts of several documents based on this technique.

The screenshot shows a web application interface for text alignment. On the left, there is a list of URNs under the heading '12 URNs'. The main area displays a table with columns for 'Document', 'Text', and 'Alignment'. The table contains several rows of text segments, some of which are highlighted in blue. The interface includes search bars and navigation controls.

Figure 14. Parallel Text Alignment Browser

The top left list contains every URN on edition level that is known to the specified CTS instance. Indents indicate groups of candidates for Candidate Alignment and can be ignored for now. After choosing the documents from the URN list at the top left, the documents are collected in the top middle list. The structure of the first document is visualized as an expandable tree view. If another document should be used as the structural template, then each can be heightened or lowered in the list using the up and down arrow. The highlighted element in the tree view is used as the text part that is supposed to be aligned. One click on the button "show table" requests the alignment for the highlighted structural element and renders it in a table. Empty text parts are included in the table and can for example indicate gaps between chapters. The alignment can also be downloaded as a tab separated text file.

Candidate Alignment only requires one CTS URN as input and uses property (2) to find suitable candidates for a set of URNs that get aligned against the passage. The last element of the {WORK} part of the given URN is deleted and then a list of fitting URNs is calculated. Then Parallel Alignment is executed with this information.

The following example uses the German translation because it includes five exemplars. The corresponding English translation can be retrieved with the URN.

urn:cts:pbcbible:parallel.eng.darby:1.7.24

and contains the text passage

And the waters prevailed on the earth a hundred and fifty days .

Using the URN

urn:cts:pbcbible:parallel.deu.luther1912:1.7.24

the last element of the {WORK} part is deleted

urn:cts:pbcbible:parallel.deu.

Then all suitable document URNs are collected

urn:cts:pbcbible:parallel.deu.elberfelder1871:
 urn:cts:pbcbible:parallel.deu.elberfelder1905:
 urn:cts:pbcbible:parallel.deu.luther1545:
 urn:cts:pbcbible:parallel.deu.
 luther1545letztehand:
 urn:cts:pbcbible:parallel.deu.luther1912:

And the passages for the following URNs are retrieved:

urn:cts:pbcbible:parallel.deu.
 elberfelder1871:1.7.24
 urn:cts:pbcbible:parallel.deu.
 elberfelder1905:1.7.24
 urn:cts:pbcbible:parallel.deu.luther1545:1.7.24
 urn:cts:pbcbible:parallel.deu.
 luther1545letztehand:1.7.24
 urn:cts:pbcbible:parallel.deu.luther1912:1.7.24

Resulting in the text alignment

Und die Wasser hatten ueberhand auf der Erde
 150 Tage .
 Und die Wasser hatten ueberhand auf der Erde
 hundertfuenfzig Tage .
 Und das Gewaesser stund auf Erden
 hundertundfuenfzig Tage .
 Vnd das Gewesser stund auff Erden hundert vnd
 funffzig tage . Mat . 24 . ; 2 . Pet . 3 .
 ; 1 . Pet . 3 .
 Und das Gewaesser stand auf Erden
 hundertundfuenfzig Tage .

Figure 15 shows a tool that aligns the individual text parts of several documents in one language based on this technique and the text variant visualisation library TRAViz ([8]).

The screenshot shows a web application interface for text alignment. On the left, there is a list of URNs under the heading '12 URNs'. The main area displays a table with columns for 'Document', 'Text', and 'Alignment'. The table contains several rows of text segments, some of which are highlighted in blue. The interface includes search bars and navigation controls.

Figure 15. Candidate Text Alignment Browser

The top left list contains every URN on edition level that is part of the specified CTS instance. The editions that are

considered as a group of candidates are indented beginning from the second candidate in the group. For example, the following candidate set for text alignment is part of the PBC CTS instance.

```
urn:cts:pbcbible:parallel.fra.davidmartin:
urn:cts:pbcbible:parallel.fra.kingjames:
urn:cts:pbcbible:parallel.fra.louissegond:
```

The list is ordered alphabetically and it is not important whether or not an entry is indented or not. Being unindented only means that it is the first one in the candidate set. The structure of the document that is selected in the top left list is visualized using an expandable tree view at the top right. Once a structural element on the lowest citation level is highlighted, the candidate alignment is requested and visualized using TRAViz. With the arrows at the side of the GUI, the next or previous neighbour text passage can be requested. The size of the span that is requested can be specified in the field "step width". For instance with the value 5 the text passage that spans the 5 next left or right neighbour URNs is requested. The image can be isolated with the "graphic" button.

Together with CTraCE, both text alignment tools are included in and deployed by the administration tool described later in this chapter.

5.2.4 Evaluation

The main disadvantage is that these methods only work if the structure markup of the documents is supporting it. Aligning a document that is structured in sentences against another one that is structured in lines does not return reliable results. If the data is prepared in a way that these methods work, they do provide two major advantages. The first advantage is that this calculation can be done in real time. The only pre calculation that is needed, is to load the data into a CTS. The second advantage follows from property (3). For most parallel text alignment methods the probability for a misalignment increases as the document continues. Each alignment comes with an uncertainty and as more text parts need to be aligned to align later text parts, this uncertainty accumulates. Because of the hierarchical structure of the {PASSAGE} parts in a CTS, the probability for a misalignment resets to its default value as soon as the border between two parent text parts is crossed. The reference for the first sentence of the fifth chapter in document A also refers to the first sentence in the fifth chapter of document B. This holds true on any citation level like for the first paragraph in the fifth chapter and the first sentence in the eighth paragraph in the seventh chapter.

The following example illustrates this effect.

```
4.8.2
4.8.3
4.9
4.9.1
5
5.1
5.1.1
5.1.2
```

After text part 4.8.3, 4.8 is exited and 4.9 is entered. 4.9.1 is now aligned without any accumulated error probability from 4.8. After 4.9.1 is exited, 5 and then 5.1 is entered. 5.1 can now be aligned without the error probability from 4. This means that misalignments in one text part do not result in misalignments in the following parts.

It is important to note that, if these methods work correctly, they work with any kind of URN except URNs that use sub passage notation like

```
urn:cts:pbcbible:parallel.eng.kingjames:35.2
@upon-35.3@my[3]
```

These URNs use the information in the text and it is unlikely that this passage will return the same result in another language because it is unlikely that the words that are used as sub notation references are located at the same position in another language if they exist at all in the specified text part.

Performance was not evaluated but is generally very good because the underlying CTS implementation performs well and the alignment is done with nothing else than a relatively small number of CTS requests. Tests showed that it is no problem to provide alignment matrices for over 30 complete bible translations in a couple of seconds.

Since it relies on the meta data markup, this method depends on the quality of the editing and assumes that the structural markup in the documents is done uniformly throughout the dataset.

5.3 Canonical Text Miner

The Canonical Text Miner is a freely available web interface that builds on the Text Mining Framework that accompanies the CTS protocol implementation ([19]) and the visualization modules that were developed for it. The purpose of this tool is to provide a graphical user interface on top of the text mining web services that hides the relatively complex process of specifying the request by URL manipulation. Figure 16 illustrates this tool using the example of its Topic Model Browser.

Figure 16. Canonical Text Miner

This tool provides two significant unique features:

- Because of the application independence of CTS URNs the results can be directly compared to other results that also use this reference system.

- Because of the underlying layers of persistent web services, each request can be stored and shared as a bookmark URL.

Text Mining techniques that are currently implemented include the mentioned Topic Model Browser, word net graphs, table N-gram viewer, neighbour cooccurrences, token frequencies and token frequency based trend analysis, and various text search methods on document and text passage granularity.

5.4 Citation Analysis

Citation analysis or text reuse is a complex task that can be accomplished in various ways. Prominent examples of text reuse analysis methods include for instance the works described in [14] and [3]. One possible generalization seems to be that all text reuse methods require some kind of reference system, a text similarity analysis and publication dates to produce results. They can be delimited from each other by the kind of similarity analysis that they use.

In the context of this work, the text passage search is used as the similarity analysis and the CTS URNs are used as references for the text passages. The publication dates are retrieved from the meta information that is part of the text inventory. The analysis process is that each smallest text part on the lowest citation level in a given CTS is used as input for the text passage search that is provided by the accompanying Text Miner and the result is considered as a set of citations. A list of stopwords can be provided. Each of the included words will be ignored when the candidate text passages are compared against the input.

While this method can be considered as superficial and too generalized, it is undeniable the use of application independent references is increasing interoperability and a successful first experiment based on the German bible translation by Luther from 1545 resulted in 2414 bible passages that were reused in 127367 text passages in the Deutsche Textarchiv similar to

passage:Am Anfang schuf Gott Himmel und Erde .
source:urn:cts:pbcbible:parallel.deu.
luther1545:1.1.1

```

urn:cts:dta:weise.ertznarren.de.norm:1352_#-
(...) herren sagte er am anfang schuf gott
himmel (...)
urn:cts:dta:justi.geschichte.de.norm:2062_#-am
anfang schuf gott himmel und erde
urn:cts:dta:seyfried.medulla.de.norm:853_#-am
anfang schuf gott himmel und erden
urn:cts:dta:hundtradowsky.judenschule01.de.
norm:750_#-am anfang schuf gott himmel und
urn:cts:dta:bullinger.haussbuoch.de.norm:13540_
#- (...) buchs im anfang schuf gott den
himmel
urn:cts:dta:luetkemann.auffmunterung2.de.
norm:8421_#-im anfang schuf gott himmel und
erden (...)
urn:cts:dta:fontane.kinderjahre.de.norm:1747
-1748_#-am anfang schuf gott himmel und
erde (...)
urn:cts:dta:fontane.kinderjahre.de.norm:1748_#-
im anfang schuf gott himmel und erde

```

```
urn:cts:dta:luther.betbuechlein.de.norm:1570-#
    _am anfang schuf gott himmel und erden
genes
```

A full evaluation is still open but the following benefits can be identified when this method is used:

- Each of the results is a *real* (re)use of the text passage because the text passage search is very strict.
- The analysis process is generalized and can be repeated with any combination of CTS instances.
- Improvements or variations can be implemented by providing an improved or adapted text passage search algorithm.
- Potentially every text corpus can be used as input as long as there is a search engine (e.g. Lucene ([12]) that supports the text language.
- As each chunk of text is processed on its own, the size of the text collections is only restricted by the text search engine that is used.

5.5 Management Tools

To comply to the usability requirement that is one of the assumed humanistic requirements, two management tools are included as part of this work: an administration backend that allows registered users to create, manage and delete CTS instances on a server and a test suite that enables users to define data specific test cases for an existing CTS instance. These management tools are described in this section and hopefully provide enough help to comply to the usability requirement.

In addition a Namespace Resolver is introduced that maps CTS namespaces to server addresses.

5.5.1 Administration Backend

Using .war file deployment, current server software like Apache Tomcat provide relatively uncomplicated means to deploy web applications. Yet to do so, users mostly have to work with a command line terminal that is probably connected by SSH and may include server specific user rights management and MySQL user management. The goal of this administration backend is to provide an uncomplicated way to manage CTS instances without requiring advanced technical knowledge about server management. Figure 17 illustrates the main interface.

[illegible]

Figure 17. Administration Backend

CTS instances can be created and the data import process initiated. The existing CTS instances on the server are listed on the left. Each instance can be individually configured, renamed, updated and managed. The "Browse Data" tab allows to get a quick overview about the data and includes the Canonical Text Reader and Citation Exporter as well as the candidate and parallel alignment tools described in this chapter. The included tools are available for embedded use and also deployed as unrestricted standalone applications. The default parametrisation of any CTS instance and license- and meta information for the text corpus can be configured in the "Servlet" menu.

5.5.2 Test Suite

As this implementation includes several features that are not covered by the official CTS specifications, it is reasonable to provide a test suite that can be used to test these functions. It is important to note that this test suite is not meant to be used as a validator for CTS but as a way to validate a user specific software or data update.

The test suite is a standalone server that can be started from any computer and test any available CTS instance against a set of locally stored test cases. The tests have to be specified similar to the following example and the expected result has to be provided in the corresponding XML file.

```
<test id="21">
  <name>GetValidReff</name>
  <description>Tests GetValidReff with level=2<
    /description>
  <expected>test -21.xml</expected>
  <request>GetValidReff</request>
  <urn>urn:cts:demo:multilang_multi:</urn>
  <parameters>level=2</parameters>
</test>
```

Tests can also be created or edited using the test suite menu illustrated in the following figure.

CTS Test Suite

Edit Test:

Name
all special characters

Description
test if special characters are recognized correctly

Request type
GetPassage

URN
urn:cts:demo:multilang_multi:1

Additional request parameters

Expected response
<GetPassage xmlns:tei="http://www.tei-c.org/ns/1.0" xmlns:cts="http://chs.harvard.edu/xmns/cts" xmlns:a="http://relaxng.org"

Save

CTS TS

Figure 18. Test Suite Test Editing

The Test Suite does not include any user management. This means that the test editing menu is publicly available. If user management is required, it has to be implemented by the server administrator.

The result of a test run is a list of failed and passed tests as illustrated in figure 19.

CTS Test Suite

Test run: Tue Sep 27 11:45:49 UTC 2016

Name	Description	Result
all special characters	test if special characters are recognized correctly	passed
error 1	Request missing one or more required parameters	passed
error 2	Invalid URN syntax	passed
error 3	Syntactically valid URN refers in invalid value	passed
GetCapabilities	Tests GetCapabilities	passed
GetValidReff	Tests GetValidReff with level=1	passed
GetValidReff	Tests GetValidReff with level=2	passed
GetValidReff	Tests GetValidReff with wrong Level	passed
GetFirstUrn	Tests GetFirstUrn	passed
GetPrevNextUrn	Tests GetPrevNextUrn	passed
GetLabel	Tests GetLabel	passed
GetPassage	Tests GetPassage	passed
GetPassage with Context	Tests GetPassage with Context	failed
GetPassage with wrong Context	Tests GetPassage with wrong Context - Error 5	passed
GetPassage - with token	Tests GetPassage with token in urn	passed
GetPassage - with token across passages	Tests GetPassage with token in urn across passages	passed
GetPassagePlus	Tests GetPassagePlus	passed

Figure 19. Test Suite Result

Previous test runs stay available for future comparisons. Differences between the expected and the received result can be further investigated as illustrated in figure 20.

CTS Test Suite

Test result: GetPassage with Context (Wed Jan 11 17:22:26 CET 2017)

Description
Tests GetPassage with Context

Result:
failed

URL:
http://ctstest.informatik.uni-leipzig.de/demo/cts/?request=GetPassage&urn=urn:cts:song:christmas_ohtennenbaum.cat:&context=2

Response:
Diff at /GetPassage: None of the elements found fully matches <request>:
Diff at /GetPassage/request: None of the elements found fully matches <requestUrn>:
Diff at /GetPassage/request/requestUrn: Expected 'urn:cts:latinLit:phi1014.phi001.lat1:1.2.1' but 'urn:cts:song:christmas_ohtennenbaum.cat:1.1.1'

CTS TS

Figure 20. Test Suite Error

5.5.3 Namespace Resolver

The purpose of the Namespace Resolver is to map CTS namespaces to server addresses and this way provide a central reliable registry. This is especially important because since the data can be copied from one CTS instance into another, it would be possible to reference a manipulated text passage with an established CTS URN. To prevent this from happening, it is advised to request CTS URNs only from trusted servers or those that are registered as the original source. Namespaces can currently be resolved similar to the following URL.

```
http://cts.informatik.uni-leipzig.de/nsresolver.html?ns=dta
```

6 Conclusion

The conclusion of this paper is that the Canonical Text Service protocol implementation that is introduced in [20] provides a competent basis for a web service for publicly available text resource retrieval. The infrastructure that is built around it includes a growing number of interoperable data sets and tools. Some of these tools introduce new methods to work with text data that are based on the fine grained reference possibilities that is introduced by CTS. As this paper shows, this infrastructure can be used to address some of the issues that are typical in the text based Digital Humanities and can provide a valuable tool set for researchers in this field.

Acknowledgements

Part of this work was funded by the German Federal Ministry of Education and Research within the project Competence Center for Scalable Data Services and Solutions (ScaDS) Dresden/Leipzig (BMBF 01IS14014B).

REFERENCES

- [1] Anne Baillot (ed.), *Briefe und Texte aus dem intellektuellen Berlin um 1800*. Berlin: Humboldt-Universität zu Berlin. <http://www.berliner-intellektuelle.eu/>
- [2] Adrien Barbaresi. *German Political Speeches, Corpus and Visualization* <http://purl.org/corpus/german-speeches>. 2012
- [3] Marco Buechler. *Dissertation: Informationstechnische Aspekte des Historical Text Re-use*. University of Leipzig. 2013
- [4] Creative Commons. *Attribution 4.0 International (CC BY 4.0)*, <https://creativecommons.org/licenses/by/4.0/>, accessed: 20161128
- [5] Alexander Geyken, Susanne Haaf, Byran Jurish, Matthias Schulz, Jakob Steinmann, Christian Thomas, and Frank Wiegand. *Das Deutsche Textarchiv: Vom historischen Korpus zum aktiven Archiv, Digitale Wissenschaft Stand und Entwicklung digital vernetzter Forschung in Deutschland*. 2011
- [6] Till Grallert (ed.). 2015. *Digital Muqtabas: An open, collaborative, and scholarly digital edition of Muhammad Kurd 'Alī's early Arabic periodical Majallat al-Muqtabas (1906–1917/18)*. <https://github.com/tillgrallert/digital-muqtabas>.
- [7] Erhard Hinrichs and Steven Krauwer. *The CLARIN Research Infrastructure: Resources and Tools for E-Humanities Scholars*. *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC*. 2014.
- [8] Stefan Jänicke, Anette Geßner, Marco Büchler, and Gerik Scheuermann. *Visualizations for Text Re-use*, *Proceedings of the 5th International Conference on Information Visualization Theory and Applications, IVAPP*. 2014
- [9] Neven Jovanovic . *Implementing Canonical Text Services in the Croatiae Auctores Latini*, University of Zagreb, <http://croala.ffzg.unizg.hr/dh2016hc-croala/2016-hc-croala-cts.html/>. 2016
- [10] Olivia Madison, John Byrum Jr., Suzanne Jouguelet, Dorothy McGarry, nancy Williamson and Maria Witt. *FUNCTIONAL REQUIREMENTS FOR BIBLIOGRAPHIC RECORDS Final Report*, International Federation of Library Associations and Institutions. 1997
- [11] Thomas Mayer, and Michael Cysouw. *Creating a massively parallel Bible corpus*, *Proceedings of LREC*. 2014
- [12] Michael McCandless, Erik Hatcher, and Otis Gospodnetic. *Lucene in Action, Second Edition: Covers Apache Lucene 3.0*, Manning Publications Co. 2010
- [13] Heike Neuroth, Felix Lohmeyer, and Kathleen Marie Smith. *TextGrid – Virtual Research Environment for the Humanities*, *The International Journal of Digital Curation*. 2011
- [14] Martin Potthast, Benno Stein, Barron-Cedeno Alberto, and Pollo Rosso. *An Evaluation Framework for Plagiarism Detection*, *Proceedings of the 23rd International Conference on Computational Linguistics, COLING*. 2010
- [15] Martin Reckziegel, Stefan Jänicke, and Gerik Scheuermann. *CTRaCE: Canonical Text Reader and Citation Exporter*, *Proceedings of the Digital Humanities, Krakow*. 2016
- [16] David A. Smith, Jeffrey A. Rydberg-Cox, and Gregory Crane. *The Perseus Project: a digital library for the humanities*, *Lit Linguist Computing*. 2000
- [17] David Neel Smith. *Citation in classical studies*, *Digital Humanities Quarterly*, 3. 2009
- [18] Manfred Thaller. *Das Digitale Archiv NRW in der Praxis*, Dr Kovac, Hamburg. 2013
- [19] Jochen Tiepmar. *CTS Text Miner - Text Mining Framework based on the Canonical Text Service Protocol*, In *Proceedings of the 4th Workshop on the Challenges in the Management of Large Corpora, LREC*. 2016
- [20] Jochen Tiepmar. *Release of the MySQL-based implementation of the CTS protocol*, *3rd Workshop on the Challenges in the Management of Large Corpora, LREC*. 2015

- [21] Jochen Tiepmar, Christoph Teichmann, Gerhard Heyer, Monica Berti, and Gregory Crane. A new Implementation for Canonical Text Services, Proceedings of the 8th Workshop on Language Technology for Cultural Heritage, Social Sciences and Humanities (LaTeCH). 2013
- [22] Jochen Tiepmar, Thomas Eckart, Dirk Goldhahn, Christoph Küras. Integrating Canonical Text Services in CLARIN's Virtual Language Observatory, To appear in: Linguistics and Literature Studies. 2017
- [23] Francois Yergeau. UTF-8, a transformation format of ISO 10646, Alis Technologies. 2003