CTS Text Miner
Text Mining Framework based on the Canonical Text Service Protocol

Jochen Tiepmar
ScaDS, Leipzig University
Ritterstrasse 9-13, 2.OG
04109 Leipzig
jtiepmar@informatik.uni-leipzig.de

Abstract

The purpose of this paper is to describe a modular framework for text mining that uses Canonical Text Service (CTS) as a data source. By combining standardized functionalities with standardized access to text data, this framework intends to reduce the heterogeneity of workflows in today's Digital Humanities and act as an important element of a text research infrastructure.
For this work the implementation of the CTS protocol described in (Tiepmar, 2015) is used. It uses advanced functionalities that are not part of the specifications of CTS. This means that, while most current modules should work with different implementations of the CTS protocol, it cannot be guaranteed that any future module will work.

Keywords: Text Mining, Infrastructure, Webservice

1 Introduction

One of the problems of text based Digital Humanities is its heterogeneity of data sources and methods. Data sources are made public in project specific ways and require the implementation of specific crawlers for each data set or a lot of manual work. Even though they are all modern projects, examples like Project Gutenberg[1], Perseus[2], Deutsches Text Archiv[3] and Eur Lex[4] each require individual ways to access the data. After the data is crawled, another problem occurs: the texts are not structured in a unified way. Each of the four examples uses a specific markup to structure their documents. DTA and Perseus offer texts in TEI/XML format, which is a text format that is often used to standardize a document's meta information. To access individual text units – for example lines – users still have to know in which way the structure is marked up in each document before being able to access it. Furthermore, if this information is not part of the TEI header, it is not possible to know, whether you should look for <l> or </lb> to access individual lines and <p> or for paragraphs. It may even be problematic to find out, how or if the document is structured in the first place.
If TEI/XML is not provided as text format, then it is already hard to divide the text from the document's meta information.
One possible solution is provided with the CTS proto-col by allowing generic access to documents and indi-rectly providing standardized access to the structure of documents. What is still missing is a collection of tools that use this access.

One of the significant features described in (Tiepmar, 2015) is CTS Cloning, which makes it possible to copy another CTS instance in parts or completely. If you can copy the text content and create a new instance of CTS, it is also possible to change it and convert the data into different formats. This paper describes the CTS Text Miner (CTS-TM), a framework with the goal of a selfsufficient, open and modular collection of methods that only require an instance of CTS as input.
Approaches to standardize text mining workflows al-ready exist and one may argue that this creates an ironic situation where a new standard is invented to solve the problem of too many standards. Yet, CTS, as it is specified in (Blackwell & Smith, 2014), was never developed as a text standard. The goal of CTS was to create a reference system based on the way that citation is done in common literature. This reference system is generic enough to be applicable to any text but on the other hand allows for exact citations as it is done for as long as literature is cited. Because of this strict and generic design, the protocol can be used as an access point for generic tools and therefore is considered as a good candidate for this work. This is also a fundamentally different approach than the ability to describe local documents in a meta format and use this meta description to convert full documents into compliant texts, as it is done for example by GATE[5].

[1] https://www.gutenberg.org/
[2] http://www.perseus.tufts.edu/hopper/
[3] http://deutschestextarchiv.de/
[4] http://eur-lex.europa.eu/homepage.html
[5] https://gate.ac.uk/

CTS requires the texts to be compliant with FRBR[6], which creates several significant limitations with re-gard to the texts that are compatible. This editorial limitation is ignored in this work that focuses on technical aspects.
Another benefit is that the URNs of CTS create a per-sistent ID system that connects text parts over multiple workflows making it possible to compare or complement results of one project with another. This also means that the ID that is returned as a result can be directly used to retrieve the corresponding text passage, which might for example be very useful for citation analysis. This persistent ID system combined with online availability can serve as a backbone for the interoperability as it is for example motivated in (Kalvesmaki, 2015).
In combination, CTS and CTS-TM make it possible to create an infrastructure where researchers can publish results and the corresponding data sets as instances of CTS and CTS-TM. Other researchers can easily reproduce the experiments by cloning the CTS instance and repeating the workflow with a given configuration of CTS-TM.

2 Requirements on CTS-TM

The target audience for this work consists of research-ers working with digital/algorithmic text analysis but includes researchers with little or no knowledge of or interest in the technical aspects of text analysis. This means that the actual work with the tools must be as easy and intuitive as possible. For these users the graphical user interface was implemented, enabling them to calculate results for the modules that are cur-rently implemented.
Since this framework is developed for a broad user base, it is hardly possible to include every important module for anyone. Especially the amount of para-metrization may vary widely or even contradict itself for different research groups. One may only need a basic default (working) parametrization for topic models while another may want to experiment with the parameters to find different results. This makes it impossible to develop individual workflows in a way that they appeal to anyone. What is possible is to design this framework in a way that users may develop their own modules or create better versions of existing ones. These modules can then become part of the main repository and this way be available for any other user as well. Creating individual modules or new workflows is something that ought to be done by users with more detailed technical knowledge, which is why this part is not fully fleshed out yet and will be completed in future work.
Language dependent algorithms often require trained models. Since CTS is not restricted to specific lan-guages, including such algorithms would require training data for any language. This means that for any of these modules and for any language that is supported, additional training files would have to be added to the package that may then not be required by individual users. To not overcomplicate things from the start, algorithms that rely on such additional resources are not added.

[6] http://www.ifla.org/publications/functional-requirements-for-bibliographic-records

Performance is currently not optimized and optimiza-tion will be part of the future work.

3 Main Architecture

This chapter describes the current state of the main architecture of CTS-TM. Since this framework and the number and kind of modules will probably grow, some of the statements may not be true for future states of the implementation.
CTS-TM is available as a .war file that can be deployed by any compliant server, like Apache Tomcat. After deployment, a .jar executable can be found in the folder WEB-INF/lib and can be started using the command "java –jar CTSTM.jar". The configuration file conf.properties is stored next to this file and can be edited to (de)activate modules or configure properties of the input and individual workflows.
The first step of the workflow is to copy the data from the specified input. For this purpose URNs that are suitable for the given configuration are collected and stored locally with the corresponding text passage and meta information. It is advised for modules to use these local files to reduce the number of HTTP requests.
After the texts are stored locally, the individual mod-ules are started sequentially. Since parallelization might get included in this framework, only one module is running at the same time, even if it may be better to start several modules in parallel.
Many modules work with single tokens. At the mo-ment, tokenization is done by JAVAs default StringTokenizer, which works well for most cases. Support for additional tokenizers, for example Lucene's tokenizers, may be included in future work.
The data storage is not fixed and module implementers may use whatever they see fit and is compatible. In the current state, MySQL is used by most modules and most data is stored in one database. Additional modules may use this connection.
The results are available via HTTP requests. It is possible to request data while it is calculated but it is advised to wait until the import process is finished.
Indexing is done at the end of each module.

4 Modules

The following modules are currently included in CTS-TM. For each module, request and import functionalities are implemented.

4.1 Term_Document_Matrix

Term_Document_Matrix creates a MySQL table where tokens are listed and counted for any document. For example the entry

Docid	Termid	Token	Count
0	45	Mit	35

describes a token “mit” which occurs 35 times in document 0.

4.2 Neighbors

The MySQL table of this module is filled similar to the table for Term_Document_Matrix, but instead of tokens, direct neighboring tokens are listed and counted. For example the entry

Docid	Id	Left	Right	Count
0	56	namhaften	Herrn	5

describes two tokens “namhaften” and “Herrn” that occur next to each other 5 times in document 0.

4.3 N_Grams

N-grams are sequences of tokens or characters with the length n. For example the sentence “The sun is shining” contains the 3-grams “The sun is” and “sun is shining”.
N is configurable and may contain multiple values that will result in a separate result set. For example the configuration “3&5” will result in one table for 3-grams and one table for 5-grams. The maximum length of one n-gram is 255, the key length of MySQL’s VARCHAR. This means that n * max_wordlength may not be longer than 255. One entry may look like

Docid	N_Gram	Count
1	aus_dem_hause	8

4.4 N_Gram_Reduce

This module creates a table for each n-gram table which contains the n-grams summed up for all docu-ments. Depending on the value for n, the number of entries is reduced significantly but the information about individual documents is reduced to the number of documents.

N_Gram	Count	DocCount
aus_dem_Hause	19	3

4.5 Term_Frequency

This module counts the occurrence of tokens in the dataset and the number of documents that contain this token. The following example describes the token “mit” that occurs 1108 times distributed over 10 doc-uments.

TokenId	Token	Count	DocCount
11994	Mit	1108	10

4.6 Term_Pruning & Document_Pruning

The purpose of these two modules is to use term fre-quency based pruning to reduce the number of tokens in several tables. Frequency based pruning is a com-mon method to reduce the number of tokens in a text corpus in Information Retrieval. The goal is to elimi-nate high frequency tokens for example to optimize text indices as it is done in (Carmel et al., 2001).
Document_Pruning only adds tokens that occur in less than a certain amount of documents. Term_Pruning adds tokens that are less frequent than a certain amount of the most frequent tokens.
For example, with the threshold 10 when using Document_Pruning, only tokens that occur in less than 90% of the documents are added. When using Term_Pruning, only terms that are less frequent than 10% of the most frequent tokens are added. The threshold for these methods can be configured.
This module requires Term_Frequency and Term_Document_Matrix as input. The results look similar to the results of Term_Document_Matrix.

Docid	Termid	Token	Count
0	106	Ewigkeit	2

4.7 Zipfian_Distribution

The module Zipfian_Distribution calculates the zipfian distribution as described in (Tullo & Hurford, 2003) for the full dataset. It requires Term_Frequency as input.

Rank	Term	Count
12	Wie	947

4.8 Stop_Words

Depending on the input that is available this module builds several tables with stop words, one for each Term_Pruning & Document_Pruning and one for the tokens of the Zipfian_Distribution with rank smaller than a configurable threshold. The stop word list based on the Zipfian_Distribution is also added as the default stopwords.txt file to be used by other modules.

Docid	Termid	Token	Count
0	45	Mit	35

Rank	Term	Count
12	Wie	947

4.9 Neighbor_Reduce

This module creates two tables similar to the table from Neighbors but without the entries which contain a left or right neighbor that is considered as a stop word by Term_Pruning & Document_Pruning.

4.10 Statistics

This table contains minimum, maximum, average, median and sum of the token count, tokens per document, document per token and terms per document.

4.11 Caching

For performance reasons some results are pre calculated and stored in local files. This includes the full token list or full zipfian distribution and other results that are expected to put a lot of load onto the database and are good candidates for data dumps.

4.12 Topic_Models

With the help of the JAVA library Mallet provided by (McCallum, 2002), topic models are calculated. The results are stored locally and in three MySQL tables and make it possible to request topics with their to-kens, topics with their documents and documents or tokens with their topics.

4.13 Document_Search_MySQL

This module stores the passages and their corresponding URNs and adds MySQL's fulltext index.

4.14 Document _Search_Lucene

This module builds a Lucene Index over all documents. The index is stored locally next to the file CTSTM.jar.

4.15 Doc_Search_Tokenlength_Signature

Instead of the text passage for the documents, this module creates a new passage that replaces the tokens with their lengths. For example the passage "The sun is shining." is indexed as "3 3 2 7 .".

4.16 Fulltext_Search

This module returns URLs for the exact passage that includes a given passage. These URLs combine the URL that is configured as input with the URN that is found. For it to work, one or many of the modules Document_Search_MySQL, Document_Search_Lucene, Doc_Search_Tokenlength_Signature or one of Term_Pruning & Document_Pruning are required as well as a CTS implementation that features fulltext search on text passage level. The required modules are used to find candidate documents. For each of these candidates, the text passage is searched by the CTS instance that is specified.
If multiple modules are configured as source for the document search, then candidate documents must be part of all their results sets.

4.17 Duplicate_Search

The most advanced module yet iterates through all the CTS URNs of every document in the CTS instance and uses their passages as input for Fulltext_Search. The goal is to create an undirected graph which connects duplicate or highly similar text passages between the input and the previously calculated data.

5 Requests

At the moment, the following requests are possible using HTTP communication. Optional [parameters] are added in round brackets.

5.1 Wordlists

- Stop word list based on [zipfian distribution | term pruning | document pruning]
- Stop words in [text] based on [zipfian distribution | term pruning | document pruning]
- [Text] minus the stop words based on [zipfian distribution | term pruning | document pruning]
- Zipfian distribution (from [rank1] to [rank2], with number of occurrences)
- Number of documents, types or tokens (for [token])
- List of documents, types or tokens (for [to-ken])
- Left or right neighbor token for [token] (plus number of occurrences)

5.2 N_Grams

- N_Grams for [n] (plus number of occurrences, up to [rank])
- documents containing [n_gram] for [n] (plus number of occurrences, up to [rank])

5.3 Search

- Documents or
- Text passages containing [text] using [mysql | lucene | term_pruning | doc_pruning | token-length_signature]

5.4 Topic Models

- Topics
- Topics plus tokens (from [rank1] to [rank2], with [weight])
- URNs for [topic] (with [weight])
- Tokens for [topic] (from [rank1] to [rank2], with [weight])
- Topics for [urn] (with [weight])

6 Evaluation

Three data sets are used for this evaluation:
1) The TED Subtitle Corpus (TED) that was published as a CTS instance in 2015 contains 52'987 relatively small documents in 105 languages. For the benchmark, only English documents were included resulting in 1938 document with 4'172'395 tokens / 56'742 types.
2) A snapshot from the text corpus "Deutsches Text Archiv" (DTA) from November 2014 as it was pub-lished as an instance of CTS. Only the normalized documents are used resulting in 1712 documents with 114'711'190 tokens / 1'565'612 types.
3) The Parallel Bible Corpus (PBC) as it was published as a CTS instance in 2015.
The test system is a virtual machine with 4 GB of memory, 6 GB Swap, a Quad-Core AMD Opteron(tm) Processor 8356 and a 350 GB ATA disk. Every request was sent via localhost and the system was rebooted before the benchmarks were started. SQL was provided by MariaDB 5.5.47 (Ubuntu 14.04.01). Apache 7.0.28 was used as the server with JAVA 1.7.0-55-b14. The default configuration was not changed. For better readability numbers are rounded to integers.

6.1 Performance

To evaluate performance, two instances of CTS-TM were created – one based on the data from the DTA CTS filtered by ".norm:" and one based on the TED CTS filtered by ".en:". For each of these instances the tokens, types, zipfian distribution and zipfian distribution from rank 2 to rank 100 were requested globally and for each document.
Table 1 and Table 2 show the number of elements and the response times for every global request. The global token list for DTA resulted in a memory error which would have to be fixed in the server configuration. The correct value is 114'711'190.

	Tokens	types	zipf	zipf2
TED	4’172’395	56’742	56’742	99
DTA	N/A	1’565’612	1’565’612	99

Table 1 Number of elements in result set

	Tokens	types	zipf	zipf2
TED	7’565	749	241	13
DTA	1’860	1’011’069	4’631	86

Table 2 Response time in MS

Table 3 and Table 4 show minimum, average and maximum number of elements and response times for every request for every document in TED instance.

	Tokens	types	zipf	zipf2
min	2	2	2	1
avg	2’153	612	612	98
max	6’618	1’362	1’362	99

Table 3 P: TED: number of elements in result set per URN

	Tokens	types	zipf	zipf2
min	11	9	9	8
avg	18	13	14	12
max	120	127	18	44

Table 4 P: TED: response time per URN in MS

Table 5 and Table 6 show results for the same bench-mark using the DTA data.

	tokens	types	zipf	zipf2
min	73	60	60	59
avg	67’004	8’228	8’227	98
max	1’082’829	51’430	51’429	99

Table 5 P: DTA: number of elements in result set per URN

	tokens	types	zipf	zipf2
min	8	8	8	8
avg	38	33	59	37
max	1865	311	845	444

Table 6 P: DTA: response time per URN in MS

The different values for zipf and types in
Table 6 might indicate an encoding related bug and require further investigation.

These results show that the performance of the system is good enough to be used productively and scales well. Both instances of CTS-TM achieve an average response time that is well below 100 MS. The impact of the higher number of tokens / types per document in DTA is not very big but measureable. DTA‘s global token list could not be requested because the string was too big to be handled by the server given the default configuration. The response times for global results will be optimized with locally stored pre calculated files (caches).

6.2 Document Search

CTS-TM provides two kinds of text search methods: document search and text passage search. Because the text passage search uses an external resource, it is not evaluated in this work. The results of the evaluation are also important for the text passage search because the document search influences the number of candidate documents that the CTS instance has to consider. Lower response times and smaller result sets have positive effects on the response times of the text passage search.

The score that is required to be considered as a candidate by Lucene is 0.1.

Evaluation was done similar to 6.1. The functions that were used are separated by the search methods that are available: lucene, term_pruning, doc_pruning and tokenlength_signature. MySQL could not be evaluated because the SQL Fulltext index was not supported by the test system. The first CTS text part of the original document was used as the query text.

	signature	termprun	docprun	lucene
min	1	1	1	0
avg	161	353	339	134
max	1938	1923	1923	1000

Table 7 TS: TED: number of elements in result set per URN

	signature	termprun	docprun	Lucene
min	8	10	9	2
avg	52	353	339	103
max	192	874	631	219

Table 8 TS: TED: response time per URN in MS

Table 9 and Table 10 show the benchmark results for the DTA data.

	signature	termprun	docprun	lucene
min	0	0	0	0
avg	340	50	50	39
max	1712	1672	1672	1000

Table 9 TS: DTA: number of elements in result set per URN

	Signature	termprun	docprun	Lucene
min	7	9	9	3
avg	720	807	769	1851
max	4957	31342	10340	21678

Table 10 TS: DTA: response time per URN in MS

Document search performs relatively well for any of the methods. That the response time for the token length signature is faster than the response time for the method using the Lucene index is surprising as are Lucene’s relatively bad response times for the DTA data set. Lucene managed to create the smallest candidate lists. However, in 253 TED requests, it did not return any result while any other method did find something.

Table 11 shows the amount of empty results for DTA.

signature	termprun	docprun	Lucene
23	1	1	412

Table 11 Number of empty results

Since every text passage was definitely part of the data set, these empty results might indicate errors.
This does not mean that Lucene performs generally worse than the other solutions, only as it is implemented in CTS-TM.

6.3 Research Value

Using Duplicate_Search a text re-use analysis was done that compared the German translations of the bible against the CTS-TM calculated with the DTA CTS with the goal to find bible citations in DTA. For this purpose, documents were filtered using the token length signature described in 4.15.
Table 12 shows the number of cited passages from PBC and citations in DTA for each of the five German bible translations that are part of PBC.

	1	2	3	4	5
PBC	32	27	361	271	57
DTA	5954	272	1667	1107	479

Table 12 Duplicate text passages between PBC & DTA
1 = elberfelder1871 3 = luther1545
2 = elberfelder1905 4 = luther1912
5 = schlachter

As it was expected, citations for bible translations by Luther were most prominent. The high number of citations of elberfelder1871 is the result of passages like “und 61000 Esel” or “und 72000 Rinder”. Numbers are deleted and “und” is considered a stop word. This results in passages like “Rinder” and “Esel”, which occur often. In elberfelder1905, these numbers are spelled in full[7].
Because of the connection to CTS, each of the citations is referenced as a CTS URN and can be used to retrieve the corresponding text passage. A simple visualization based on a prototype of (Reckziegel et al., 2016)’s CTRaCE uses this connection to list each citation with links to the corresponding text passage in DTA:

Am Anfang schuf Gott Himmel und Erde .
urn:cts:pbc:deu.luther1545:1.1.1
-- am anfang schuf gott himmel und erde
-- im anfang schuf gott himmel und erde[8]

Furthermore, since PBC is a parallel corpus, the URNs can be used to align the citations over different translations in one language, for example resulting from different use of metaphors. The text passage "ich wache, und bin wie ein einsamer Vogel auf dem Dache." from elberfelder1871 was found as a citation in DTA. The corresponding text passage "Ich bin gleich wie eine Rohrdrommel in der Wüste; ich bin gleich wie ein Käuzlein in den verstörten Stätten." in luther1545 was not found but can be associated when the results for the different translations are aligned using the alignment of CTS URNs. Furthermore, this means that any translation of the bible that is part of PBC can potentially be aligned against these results, making it possible to create citation graphs based on DTA for each of the 831 translations. And finally, because every result that is calculated with CTS-TM shares the same CTS URNs as identifiers, these results can be enriched with any other result of CTS-TM or tool that is developed with compatibility for the CTS protocol.
The results show high precision but low recall. High precision can be implied because each citation is an exact reference to this text passage in DTA if stop words and numbers are ignored. Recall is hard to measure because there does not exist a complete list of bible citations in DTA. DTA includes many phrases with variations of single tokens. For example, the phrase "Am Anfang schuf Gott Himmel und Erde." also occurs as "Am Anfange schuf Gott Himmel und Erde." or "Am Anfange schuff Gott Himmel und Erde.". These variations are not included in the results but can be covered by including editing distances in the fulltext search.
Another issue is based on the format of PBC. Duplicate_Search uses the smallest referencable CTS text parts as input. These may be too big for certain famous text passages like "Nehmet, esset, dies ist mein Leib", which appears nowhere in the results. However, when searched explicitly, this phrase is found 932 times in DTA. The problem is that this phrase is always included in bigger contexts like "Da sie aber aßen , nahm Jesus das Brot , dankete und brach's und gab's den Jüngern und sprach : Nehmet , esset ; das ist mein Leib .", which do not appear in DTA. Segmentation techniques can be applied to divide phrases into smaller text parts but this would require additional language dependent resources. The easiest way to include such phrases is to create an additional edition with this in mind and use it as input.
The resulting citation graph is not directed, which is not a problem in this case. It is unlikely that the bible re-used a passage from DTA. For future workflows the publication dates that are available as meta information in CTS can be used to create directed graphs.
Much more focused work concerning text re-use was done by (Büchler, 2013) or can be done with the help of Winnowing described in (Schleimer & Wilkerson & Aiken., 2003). The goal of this evaulation was to illustrate the benefit of the interoperability that is provided by the shared set of identifiers by combining an algorithmic workflow of text re-use with an algorithmic workflow of text alignment and a generic visualization tool.
As proof of concept, the binaries, source code and configuration for CTS-TM, as it was evaluated in this work, are available online[9] so that anyone can repeat the evaluation locally.

[7] See urn:cts:pbc:deu.elberfelder1905:4.31.34 and urn:cts:pbc:deu.elberfelder1905:4.31.33

[8] Complete results for luther1545: http://ctstm.informatik.uni-leipzig.de:8080/tr_dh/

[9] http://ctstest.informatik.uni-leipzig.de/eval/ctstm_release.zip

7 Conclusion

As it is clearly shown in 6.3, the connection of CTS with standardized workflows shows a lot of potential.
Especially the combination of separate results and tools by the usage of a shared set of identifiers and the fact that all of the results can be easily recreated using data that is publicly available, creates a transparent and interoperable environment. When working with parallel text corpora, the research results can even be shared across language barriers.
Future Work might include the implementation of additional or alternative modules and the addition of generic visualization modules, for example for bags of words, links between documents or topic models. Additional tokenizers can be implemented to enhance import functionalities. Since Lucene is already part of this framework, implementing compatibility with Lucene's tokenizers might be the easiest way to do this while making sure that future state of the art tokenizers can be included without much implementation effort.
Additionally, CTS-TM must be further connected to established text workflow frameworks and document databases like Sketch Engine, GATE and other text mining oriented data storages as it was already done with Lucene and MySQL's fulltext index. The connection to workflow orientated tools like KNIME might also create a lot of opportunities for interoperability between different tools.

8 Acknowledgements

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)

9 Bibliographical References

Blackwell, C. & Smith, N. (2014). Canonical Text Services protocol specification. Retrieved from http://folio.furman.edu/projects/citedocs/ctsurn/ and http://folio.furman.edu/projects/citedocs/cts/ 2015, February 19.

Büchler, M. (2013). Informationstechnische Aspekte des Historical Text Re-use (English: Computational Aspects of Historical Text Re-use). PhD Thesis. Leipzig University.

Carmel D. & Cohen D. & Fagin R. & Farchi E. & Herscovici M. & Maarek Y. & Soffer A. (2001). Static Index Pruning for Information Retrieval Systems. In Proc. ACM SIGIR.

IFLA Study Group on the Functional Requirements for Bibliographic Records (1998). Functional Requirements on bibliographic records: final report. In UBCIM publications; new series, vol. 19. München, K.G. Saur.

Kalvesmaki J. (2015). Three Ways to Enhance the Interoperability of Cross-References in TEI XML. In Proceedings of the Symposium on Cultural Heritage Markup. Balisage Series on Markup Technologies, vol. 16.

McCallum, A. (2002) MALLET: A Machine Learning for Language Toolkit. http://mallet.cs.umass.edu. 2002.

Reckziegel, M., Jänicke S., Scheuermann G. (2016). CTRaCE: Canonical Text Reader and Citation Exporter. To appear in Proceedings of the Digital Humanities, Krakow.

Schleimer S. & Wilkerson D. & Aiken A.(2003). Winowing: local algorithms for document fingerprinting. In Proc. Of the 2003 ACM SIGMOD Intl. Conf. on Management of data, pages 76-85.

Tiepmar J. (2015) Release of the MySQL based implementation of the CTS protocol. In Proceedings of the 3rd Workshop on Challenges in the Management of Large Corpora (CMLC-3).

Tullo C & Hurford J. (2003). Modelling Zipfian Distribution in Language. In Kirby, S. Language Evolution and Computation, Proceedings of the workshop at ESSLLI.