Final Report Karl-Steinbuch-Stipendium 2013/14

Gesa Stupperich, Qiu Zhiqiong, David Höppner and Manuel Sassmann

November 3, 2014

Abstract

In the main part of this article we give a technical overview of our platform for automatic and manual TEI annotation of classical Chinese texts from the *Siku quanshu*. We conclude with a section dealing with the preliminary results of further research directions and give suggestions for further development.

1 Introduction

The *Siku quanshu* 四庫全書 (Complete library of the four branches) was one of the last major imperial editorial project of the late imperial China.[1] The sixth ruler of the Qing 清 dynasty, the long reigning Qianlong 乾隆 emperor (r. 1735–1796) issued an edict, on the very first day of the new chinese year 1772, calling on his officials to collect books from all over the empire to expand the imperial book collection. In the course of the edition process under the chief editors Ji Yun 紀昀 (*zi* Xiaolan 曉嵐, 1724–1805) and Lu Xixiong 陸錫熊 (*zi* Jiannan 健男, 1734–1792) 3471 works (in 79070 *juan*) of the 13501 works sent in where selected for inclusion. For additional 6793 titles extensive bibliographic and scholarly notes where written and published

[1] The following section is based on the excellent description found in Endymion Wilkinson, *Chinese History: A New History* (Cambridge (Massachusetts): Harvard University Press, 2012), 945-954. For the political background, see R. Kent Guy, *The emperor's four Treasuries: Scholars and the State in the late Ch'ien-lung Era* (Hong Kong: Hong Kong University Press, 1987)

under the title *Siku quanshu zongmu tiyao* 四庫全書提要 (Catalogue (with critical abstracts) of the Complete library of the four branches).

The phrase *siku* 四庫 in the title of the *Siku quanshu* denotes the four traditional categories *jingbu* 經部 (Classics), *shibu* 史部 (Histories), *zibu* 子部 (Philosophy) and *jibu* 集部 (Literature) of chinese bibliographic sciences that emerged in the 3rd century. Those four categories express an value ordering of different genres of chinese literature.

We chose the follow texts for our exemplary TEI annotation work. Two texts are annotated by hand, the rest was automatical annotated by the UIMA backend.

1. The *Shen xian zhuan* 神仙傳 (Traditions of Divine Transcendents) collects hagiographies of transcendents beings. Its the second major work by Ge Hong 葛洪 (*zi* Zhichuan 稚川, 283–363) besides his more famous *Bao puzi* 抱朴子 (Master Who Embraces Simplicity). The text contains special vocabulary often found in daoist sources. Including plant, animal and mineral names as well as famous recipes. Annotated by hand.

2. The *Qiusheng ji* 秋聲集 (Sounds of Autumn Collection) gathers the poetry of the Yuan dynasty literatus Huang Zhencheng 黄鎮成 (*zi* Fuzhen 符鎮, 1288-1362) into a 3 *juan* work. It was chosen to demonstrate and to experiment with rhyme and tone annotations. Annotated automatically with UIMA components.

3. The 宋史 (History of the Song [dynasty]) is the most voluminous official history *zhengshi* 正史 included in the *Siku quanshu.* Compiled between 1343–1345 under the chief editor Toghto Tuotuo it is a vast source for the nomenclature of the administrative aspects of the imperial rule. Annotated automatically with UIMA components.

4. The 天主實義 (True Meaning of the Lord of Heaven) and 天問略 (Epitome of Questions on the Heavens) include technical terms for argumentation and anstronomy. Annotated by hand with citable page references of the base-text used for digitization.

2 Technical Report

2.1 Deployment and Development Environment

In the orginal project proposal we intended to use OpenIndiana[2], an illumos[3] based distribution, as our base deployment and development platform. While we still consider it a better platform with the availability of better debugging and analysis tools such as dtrace and mdb, and a better filesystem in the form of zfs, we did run into problems connecting various shared-ip zones over a single network interface. As we didn't want to waste too much critical time at the starting period of the project, we switched to a linux based distribution and used containers (LXC) to separate our services. For the container management we initially used pre-stable release of docker.[4] That was a slightly bumpy road, as we encountered various bugs and interface changes. With the advent of docker version 1.0 the interface has stabilized and the platform has matured. But its design is still inferior to other container implementations like the solaris zones or freebsd jails.

We provide docker container build recipes for our postgres, exist, fuseki, rabbitmq and django services. For the developer we provide a vagrant[5] file, which installs our deployment distribution (CoreOS) into a virtual machine and then provisions our docker containers.[6] Thus the developer only needs to run the command # vagrant up to install a local development environment. Deployment to a server is similarly simple by running the docker container build recipes by hand.

2.2 Web Application

The web application provides the primary interface to the user. Its the main interaction point with our services. The web application connects and interacts with the various data sources and presents the results to the user. We currently use the django framework[7] with various additional modules, the main advantage of a python based framework resides in the availability

[2] http://openindiana.org/

[3] http://wiki.illumos.org/display/illumos/illumos+Home

[4] https://www.docker.com/

[5] https://www.vagrantup.com/

[6] CoreOS is a custom minimal distribution just to run containers https://coreos.com/ CoreOS its similar to the illumos based SmartOS.

[7] https://www.djangoproject.com/

of some third-party modules, that ease the interaction with existdb and XML documents.[8] The user is presented with an interface to browse our collection of texts by author or title. Other services are also available to the user, like our OCR or the annotation service.

We consider the git file repository of our TEI documents as the master copy of our documents. The main reason for this stipulation resides in the UIMA component which annotates new entities on demand, but for which we currently only have a file collection reader. If an update to the master copy is applied, our XML database will reload the documents that did change.

2.2.1 Text Viewer

The user can browse TEI documents currently loaded in the XML database by title or author. If s/he desires to view a single document, the TEI document under question will be fetched from the XML database. Before the document is presented to the user it must be XTSL transformed to rewrite the TEI tags into valid HTML tags.

Texts and single chapters can be downloaded in a plain text format or as pdf files. The user has the option to undisplay various annotations he has no interest in. For example he can hide the (modern) punctuation by pressing ”.” (full stop). Users can consult the help page for further keyboard shortcuts.

New annotations can also be added in the viewer. The user selects the phrase he wants to mark with the mouse and then enters a keyboard shortcut to tag the selection. The marked phrase is sent to the server and after validation enters our RDF store as a new fact. Our RDF store is partitioned into 2 named graphs. New facts supplied by users are by default stored in an extra graph, to seperate them from the core verified RDF store. A list with the available keyboard commands can be found within the documentation section of the website. Annotations can be viewed and managed in the django admin interface if they are not automatically added to the RDF store.

2.2.2 OCR

Part of our web application is an OCR service. We use tesseract[9] in the backend for the actual recognition. As tesseract is an open source solution the

[8] Specifically two packages from Emory Libraries: eulxml and eulexistdb.

[9] `http://code.google.com/p/tesseract-ocr/`

accuracy does not reach the levels of commercial offerings. Our experiments with ABBYY FineReader (Version 12) resulted in a much better overall recognition rate, but the licensing fee is bound to the pages scanned.[10] We hope the Internet Archive [11] upgrades they version of ABBYY to provide better full text versions of their PDF scans.

2.3 SPARQL

We also run a public SPARQL (**S**PARQL **P**rotocol **A**nd **R**DF **Q**uery **L**anguage) endpoint. This endpoint can be used via a web interface or programmatically through a client library. It provides access to our manually modeled facts and also to facts extracted from automatic annotated sources. We use the functional notation of OWL (Web Ontology Language) to generate the corresponding RDF (Resource Description Framework) triples. Currently we employ a java based RDF store.[12] If this solution does not scale well enough we might evaluate 4store as an alternative.[13]

2.4 UIMA

UIMA (Unstructured Information Management Architecture) started as an IBM project (Watson).[14] It was later open sourced and is now an apache project. UIMA is an architecture and a standard for annotating and enriching unstructured data. It can be used for textual sources, but it is not limited to this domain. In UIMA various components interact on top of a common analysis structure (CAS) to annotate documents. At the core we find analysis engines (AE) that annotate single documents by inserting types with offsets into the CAS. Analysis engines can share common knowledge (data structures) by using shared resource objects. This not only bundles source code into a single place but also speeds up processing as common data structures will be constructed only once and will be cached between single documents and by all engines. Before the analysis engines analyse documents, those documents must be transformed into the CAS format. This is done by the collection reader interface. Collection readers iterate through a document

[10] `http://www.abbyy.de/`

[11] `http://www.archive.org/`

[12] `http://jena.apache.org/documentation/serving_data/index.html`

[13] `http://4store.org/`

[14] `http://uima.apache.org/`

collection and build CAS structures for every single element. The CAS is then passed on to the analysis engines. As a final step the CAS can be serialized again by a CAS consumer to generate a document in the original document format. In our UIMA component we follow this workflow but an additional CAS consumer extracts new information from the annotation it encounters in the newly processed document. As analysis engines can be run in parallel or sequential order, as independent or depend engines (engine B needs the annotation results from engine A), schedules are needed. We currently use a simple linear pipeline to run our analysis engines. UIMA currently provides only this functionality. More complex setups need additional scale out frameworks that are not as easy to deploy.

While UIMA provides a mature and well structured architecture for text analysis, some features contrain the practical use.[15] In our context the static type system has the greatest negative impact. As we can hold to the precondition that all the input documents to our collection reader are valid TEI documents, we would preferable create for every encounted TEI tag a corresponding UIMA type and an annotation. This would allow us to preserve all TEI tags encounted, even if the user does not use them himself in his analysis engines. With a static type system as currently present in UIMA we need define all tags beforehand. As the TEI standard is quite extensive in the number of tags introduced, this is a tedious and laborious task. Therefore we currently do not implement the whole standard but only a selection of tags our analysis engine use in their automatic annotations.

2.5 Future Improvements

2.5.1 Infrastructure

Single services should be clustered to provide better availability. While CoreOS's key-value store etcd and fleet provide such capabilities, Mesos[16] or Kubernetes[17] are likely better solutions.

[15] Thilo Götz, Jörn Kottmann, and Alexander Lang, "Quo Vadis UIMA?," *Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT* (Dublin):77–82.

[16] `http://mesos.apache.org/`

[17] `https://github.com/GoogleCloudPlatform/kubernetes`

2.5.2 Web Application

A full XML database is not necessary for the main functions of the site. Documents should be stored in a key value cache to provide better response times and to simplify the core functions.

2.5.3 UIMA

Besides the java version UIMA also provides a C++ interface. A possible speed up of the interactive markup should be evaluated if we rewrite the manual annotator in C++ and trim it down to its basic functionality.

3 Date and Time Expressions

In the following section we will outline a concrete annotation example. Date time expressions are particular handy, as they play a major role in historical documents and occur in a fixed standard format.

3.1 Dynasties

In our context a dynasty is a period of time in which a succession of hereditary rulers by patrilineal descent from the dynastic founder ruled.[18] The probably most famous Chinese dynasty known to the general readership is the Tang 唐 dynasty which was ruled by the Li 李 clan and lasted from 618 AD to 907 AD. An overview of all terms for imperial dynasties can be found in the literature.[19] Most expressions for dynasties in chinese are polysemes. For example the aforementioned *tang* has also the meanings of 'dike' or it can function as a place name.[20] This introduces the general problem of ambiguous expressions, which can only be solved by further analysis of the expression itself or/and by more context. This is a problem which solves itself for most expressions with more then two characters, as the percentage of ambiguous words drops dramatically. So the expression *Nan Tang* 南唐 and *Hou Tang* 後唐 are less ambiguous. Even more certain are expressions like *Tang chao* 唐朝 and *Hou Tang shi* 後唐時 that carry a sematic marker of a time expression.

[18] Wilkinson, *Chinese History*, 3.

[19] Table 3 in ibid., 4

[20] Wang Li, *Wang Li guhanyu zidian* (Beijing: Zhonghuo shuju, 2000), 117.

If we observe the contextual occurences of single character dynasty expression we find the following uses:

3.1.1 Dynasty + Person

A dynasty appears in prefix position before a person's name to qualify the person further.

Table 1: Dynasty expression before person name

Tang Han Yu	唐韓愈	Han Yu (768-824) of the Tang [dynasty]
Tang Xuanzong	唐玄宗	Emperor Xuan of the Tang [dynasty]
Tang shushi Su Jiaqing	唐術士蘇嘉慶	esoteric art expert Su Jiaqing of the Tang [dynasty]
Tang xiang Pei Du	唐相裴度	chancellor Pei Du (765-839) of the Tang [dynasty]

We iterate over persName annotations and check for prefix dynasty expressions.

3.1.2 Dynasty + Expression

A certain part of the particular dynasty is denoted.

Table 2: Dynasty as part of a time range

Song xing qie bainian	宋興且百年	the Song [dynasty] flourished since hundred years
Song zhi zhongye	宋之中葉	in the middle period of the Song [dynasty]
zi Tang qi jin	自唐迄今	from the Tang [dynasty] till today
zi Tang zhi Zhou	自唐至周	from the Tang [dynasty] till the Zhou [dynasty]

We search for fixed expressions and mark them.

3.1.3 Dynasty + Era

Starting in the Han dynasty single reign periods where divided into eras. Usually the era name is followed by an expression which marks the begin, middle or end of the era.

We search for the limited number of era names in a document. In a second pass we resolve dynasty names before them.

Table 3: Dynasty with era names

Tang wude chu	唐武德初	at the beginning of the Tang era *wude*
Tang kaiyuan zhong	唐開元中	in the era *kaiyuan* of the Tang
Tang kaiyuan mo	唐開元末	at the end of the *kaiyuan* era of the Tang
Tang kaiyuan jian	唐開元間	in the *kaiyuan* era of the Tang

Table 4: Dynasty with era names

Tang tianbao wu nian	唐天寶五年	the fifth year of *tianbao*

3.1.4 Dynasty + Era name + Year

Again the era name is the most specific component. Iterate over all era names in a document and look for year measure suffixes.

3.1.5 Full Dates

If the dates are extended to the day level and behind we arrive at more complex expressions that mix different counting systems.

Having annotations for dates leads to various additional annotations that could be done. This could be explicit events like eclipses of the sun or droughts that are regulary mentioned in texts. But also implicit cultural information like certain lucky days chosen for traveling or major undertakings.

24 Solar Terms

4 General Problems

We looked at various external sources to obtain input data for our UIMA annotators. While the CBDB (China Biographical Database) Project[21] provides a extensive data set for persons between the 7th and and 19th century, no indication of the relative importance of a person is given. Importing all 328000 records for individuals would slow down the annotation process, while inserting many possible wrong annotations. To provide a more controlled development environment for our annotation components, we postponed the import of this data set. There are open scalability issues with our current

[21] http://isites.harvard.edu/icb/icb.do?keyword=k16229

RDF store that need to be solved before. We wrote some code to convert the SQL data provided by CBDB into OWL.

More in line with our own use of RDF as the main data representation stands the DBpedia project.[22] The DBPedia project generates RDF triples from the data found in info boxes that occur in Wikipedia articles. The generated triples can be downloaded or queried online via a SPARQL endpoint. In general Wikipedia provides a more narrow set of well known individuals, but the included info boxes, which are vital for dbpedia, are distributed very unevenly among the different language versions. In the Chinese version, most relevant for our purposes they are absent in most cases. At present too many entries are without info boxes and the individual is missing in the generated triple data set.

5 Conclusion and Future Development

Automatic annotations offer possibilities for information extraction and advanced searches. As a trivial example we can now abstractly search for the biggest numeral contained in a text. Or study the distribution of surnames in a number of texts over a long historical period to gain insight into migration patterns.

Easily accessible and parseable data of Chinese texts allows for new uses of data science and data visualization workflows.

The website will also be extended by two additional analysis tools in near future. These tools already work as stand-alone version,[23] but could not be integrated into the web framework so far due to compatibility problems. One is a tool for the visualization of intertextual references. In classcial Chinese prose and poetry, explicit quotations and implicit references to other authors and texts are common stylistic devices. The tool enables the user to compare a source text against the complete Siku Quanshu or parts of it in order to identify identical or quasi-identical passages. Passages in the source text that contain parallels within one or multiple texts of the Siku Quanshu corpus are marked in color. After the tools will be integrated, the user will also be able to choose a marked passage and list all its parallels and view them in their original context.

[22] http://dbpedia.org/About

[23] https://github.com/neinkeinkaffee/siku

The other planned feature is functionality for displaying topic models of the Siku Quanshu, either for the complete corpus or for parts of it. Topic modelling is a machine learning method that automatically divides the vocabulary of a corpus into semantic clusters (topics). Topic modelling is employed in the library sciences for automatic categorisation of books. There have also been attempts in historical and literary research to employ topic modelling for gaining an overview of the intrinsic structure of large amounts of texts.**Templeton:2011** Open source implementations exist for the topic modelling algorithms described in **Dirichlet:2003** The direct output of the algorithms and the tools, however, are the numeric parameters of a probabilistic model. In order for the user to make sense and evaluate these parameters, he has to be able to summarize and visualize them. The following three basic display methods will become part of the web framework: (i) display of the semantic content of the indiviual topcs by listing their most characteristic words; (ii) display of the polysemy of invividual words by listing the topics which the respective word characterizes; (iii) display the topics of an individual text by listing its most prominent topics and by annotating each word of the text with its topic.