Chinese Text Project: A Dynamic Digital Library of Pre-modern Chinese[*]

Donald Sturgeon[†]

Abstract

This paper presents technical approaches and innovations in digital library design developed during the design and implementation of the Chinese Text Project, a widely-used, large scale full-text digital library of pre-modern Chinese writing. By leveraging a combination of domain-optimized Optical Character Recognition (OCR), a purpose-designed crowdsourcing system, and an Application Programming Interface (API), this project simultaneously provides a sustainable transcription system, search interface and reading environment, as well as an extensible platform for transcribing and working with pre-modern Chinese textual materials. By means of the API, intentionally loosely integrated text mining tools are used to extend the platform, while also being reusable independently with materials from other sources and in other languages.

Introduction

Traditional full-text digital libraries, including those in the field of pre-modern Chinese, have typically followed top-down, centralized, and static models of content creation and curation. In this type of model, written materials are scanned, transcribed by manual effort and/or Optical Character Recognition (OCR), then corrected manually, reviewed, annotated, and finally imported into a system in their final, usable form. This is a natural and well-grounded strategy for design and implementation of such systems, with strong roots in traditional academic publishing models, and offering greatly reduced technical complexity over alternative approaches. This strategy, however, is unable to adequately meet the challenges of increasingly large-scale digitization and the resulting rapid growth in potential corpus size as ever larger volumes of historical materials are digitized by libraries around the world.

The Chinese Text Project (https://ctext.org) is a full-text digital library of pre-modern Chinese written materials which implements an alternative model for creation and curation of full-text materials, adapting methodologies from crowdsourcing projects such as Wikipedia and Distributed Proofreaders (Newby and Franks 2003) while also integrating them with full-text database functionality. In contrast to the traditional linear approach, in which all stages of processing including correction and review must be completed before transcribed material is ingested into a database system, this approach works by immediately ingesting unreviewed materials into a publicly available, managed system, within which these materials can be navigated and used, as well as improved through an ongoing collaborative correction and annotation process. From a user perspective, this has the consequence that the utility of the system does not rest upon prior expert review of materials, but instead derives from provision to individual users of the ability to interact directly and effectively with primary source materials and verify accuracy of transcription and annotation for themselves. Combined with specialized Optical Character Recognition techniques leveraging features common to pre-modern Chinese written works

[*] This paper has been accepted for publication in *Digital Scholarship in the Humanities*, version of record: https://doi.org/10.1093/llc/fqz046

[†] Department of Computer Science, Durham University.

(Sturgeon 2017a), this has enabled the creation of a system providing access to a long tail of historical works which would otherwise not be available in transcribed form. The system is highly scalable and currently contains over 25 million pages of primary source material and over 5 billion characters of transcribed text, while being used by over 30,000 users around the world every day.[1]

As the scale of the project has grown considerably since originally envisioned, many opportunities for text mining and other digital studies using material contained within the platform have emerged. To facilitate these uses, an Application Programming Interface (API) has been implemented, which is currently used for text mining purposes in research and teaching at a number of universities worldwide, as well as for facilitating data exchange with independently developed online tools, such as the MARKUS textual markup platform for historical Chinese texts developed at Leiden University (De Weerdt et al. 2016) and the Text Tools textual analysis platform (Sturgeon 2019, forthcoming).

Basic functionality and representations of data

Systems such as MediaWiki have been used successfully to enable crowdsourced collaboration of the transcription of historical textual works – a notable example being the Wikisource project (https://wikisource.org), which allows anonymous and pseudonymous users to contribute to the transcription of texts in a large range of languages. While the quality of transcriptions in Wikisource varies greatly, open collaborative transcription systems have been shown to be capable of producing accurate transcriptions (Neudecker and Tzadok 2010). Transcriptions of texts created as part of Wikisource have been widely used in text mining and Natural Language Processing (NLP) applications, as have those produced through other crowdsourced transcription platforms such as Distributed Proofreaders.

One key difference between the system described in this paper and many crowdsourcing transcription projects is that in the Chinese Text Project, collaborative editing is in general expected to be an ongoing, continual process, rather than one which should be completed prior to use of fully corrected transcriptions. The extremely large volume of primary source material available – the project currently provides access to over 25 million pages of scanned material – means that even with a large and committed crowdsourcing community, many more obscure materials cannot be expected to be manually corrected for some time. As a result, the *use* of these materials is not expected to be something which can take place only after correction has been completed; instead, use may occur at any time starting from the initial creation of a transcription by OCR. Full-text search of scanned image sequences, full-text indexing of the entire body of material, extraction of full-text data in a readable form, and API access to content are all expected to take place at any point in the process of correction, from first ingestion of unreviewed data through to completed correction. This dynamic model contrasts with both the traditional static model of content curation for full-text digital libraries, as well as with models such as those of Distributed Proofreaders and other systems implicitly relying upon a two-step process of complete correction in a system designed solely for transcription, followed by use of fully corrected materials in separate systems not implementing crowdsourced correction (Fig. 1).

This design requirement has important consequences for the data structures used to represent textual materials during the crowdsourcing process. For example, one serious difficulty faced in using data contained in MediaWiki-type systems such as Wikisource for text mining purposes is the high degree of freedom of structure allowed by the page-based or “article” model.

Software like MediaWiki allows the editing of individual pages or "articles" (directly analogous to encyclopedia articles in Wikipedia, which is built using the same software platform), which for practical reasons should not be arbitrarily long. In Wikisource, a single long text is frequently represented by using a separate page for each chapter of the text, together with another page containing links to each chapter in sequence. However, neither the MediaWiki platform nor the Wikisource implementation enforce this particular choice of structure, and many other structures are both possible and observed in practice – for example, multiple levels of contents pages, as well as composite texts which are themselves composed of other texts. More seriously, as the platform does not enforce rules on which pages may link to which others, the link structure that would need to be navigated to serialize a complete text, starting from its contents page, is modeled as an arbitrary directed graph – allowing cases such as cycles as well as disconnected or "orphan" pages. Other complicating factors include the presence of multiple links intended for human readers (such as navigation bars), which would need to be specifically excluded by an indexing system attempting to construct the entire text as a single object by identifying and assembling all of its various pieces in the correct order.

Page-based models of representation in which pages correspond to physical pages in a source manuscript easily sidestep this issue, because pages occur in sequence and so it should be possible to assemble their contents in the correct order with little difficulty; this type of approach is common in dedicated transcription systems in which a scanned source existing in the system is a prerequisite to any transcription being created.[2] In this approach, division into logical organizational units such as chapters is no longer a core feature of the representation; though in principle this could be added using markup conventions, the currently used markup in projects such as Distributed Proofreaders does not provide markup to indicate this, instead asking proofreaders to use markup describing the *formatting* of a chapter heading.[3]

The representation chosen for the Chinese Text Project attempts to combine aspects of both approaches in such a way as to record information about physical page locations as well as logical hierarchies in the text such as chapter divisions, while also constraining the maximum size of internally represented textual units and minimizing the required complexity of processing for display of textual content, indexing, and assembling of complete texts. The representation is based on two layers: firstly, a complete text is represented as a single sequence of one or more serialized textual items of sizes limited to those which can be practically edited in a web browser window. These items are internally referred to as "chapters" (though they need not always correspond to chapters in the text), and correspond directly to the top-level divisions presented to a user when navigating the text. Secondly, each "chapter" consists of a serialized fragment containing the full-text content of that item, together with additional information that 1) links every line of text to a position on the particular page image from which the transcription derives, and 2) encodes logical subdivisions within the "chapter", including paragraph divisions. Fragments contain XML markup together with a small number of additional markup conventions, most notably the convention that a blank line is interpreted as a paragraph division. This convention was chosen because it has the effect that a text not containing any markup (thus representing a transcription which has no links to a scanned primary source) is both a valid serialization and also has an easily readable internal representation. This representation allows the system to perform versioning at the level of the "chapter", which is of a controlled length; it ensures that when displaying a text one chapter at a time, only one underlying unit of serialized data needs to be processed; and it provides a simple way of constructing the complete text, by assembling the list of units in the specified order.

The key function of the chosen representation is to facilitate and connect two distinct methods of interacting with the transcribed material: firstly, as a single document consisting of the transcribed contents of each page concatenated in sequence to give readable plain-text with logical structure (divisions into chapters, sections, and paragraphs); secondly, as a sequence of page-wise transcriptions, in which a direct visual comparison can be made between the transcription and the image from which it is derived (Fig. 2). In both cases, an important contribution of the transcription is that it enables full-text search; the primary utility of the page-wise view is that it enables efficient comparison of transcribed material with the facsimile of the primary source itself (Fig. 3). As these two views are linked to one another and created from the same underlying data, this makes it feasible to read and navigate a text according to its logical structure, and at any stage of the process jump to the exact corresponding location in the sequence of page images to confirm accuracy of any part of the transcription.

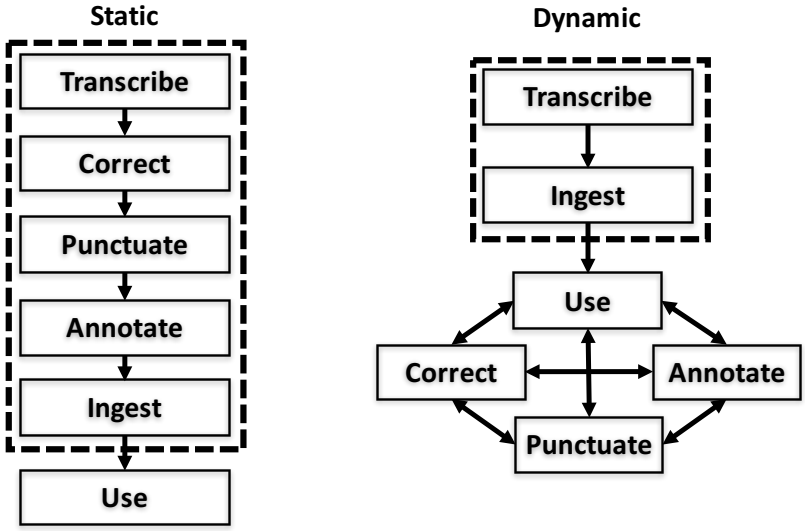
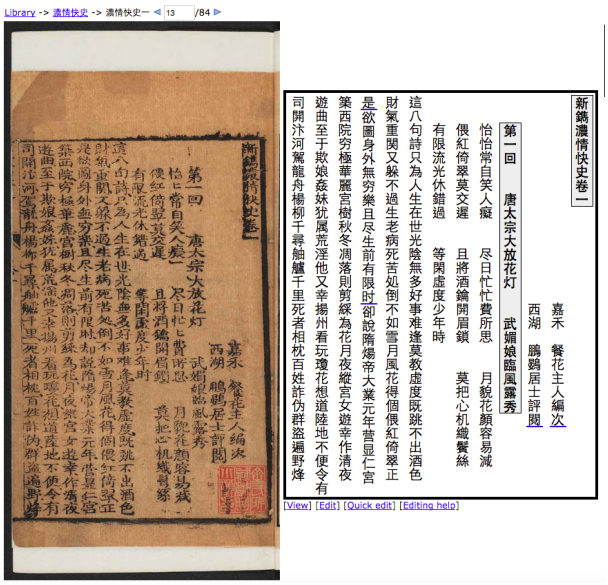


Figure 1. Static and dynamic models of full-text database content curation. Dotted regions indicate parts of the process occurring prior to ingestion into the final platform, during which results are inaccessible to users.



新鐫濃情快史卷一

第一回　唐太宗大放花灯　武媚娘臨風露秀

嘉禾　餐花主人編次

西湖　鵑鵑居士評閱

怡怡常自笑人癡　尽日忙忙費所思
月貌花顏容易減　偶紅倚翠莫交遲
且將酒鑰開眉鎖　莫把心机織鬢絲
有限流光休錯過　等閑虛度少年時

這八句詩只為人生在世光陰無多好事難逢莫教虛度既跳不出酒色財氣重関又躲不過生老病死苦処倒不如雪月風花得個偶紅倚翠正是欲圖身外無窮樂且尽生前有限時卻說隋煬帝大業元年營昆仁宮築西院穷極華麗宮樹秋冬凋落則剪綵為花月夜縱宮女遊幸作清夜遊曲至于欺娘姦妹猶属荒淫他又幸揚州看玩瓊花想道陸地不便令有司開汴河駕龍舟楊柳千尋舳艫千里死者相枕百姓詐伪群盜遍野烽

[View] [Edit] [Quick edit] [Editing help]

《新鐫濃情快史卷一》

1 嘉禾餐花主人編次
2 西湖鵑鵑居士評閱

《第一回唐太宗大放花灯武媚娘臨風露秀》

1 怡怡常自笑人癡，尽日忙忙費所思。
月貌花顏容易減，偶紅倚翠莫交遲。
且將酒鑰開眉鎖，莫把心机織鬢絲。
有限流光休錯過，等閑虛度少年時。
這八句詩，只為人生在世，光陰無多。好事难逢，莫教虛度。既跳不出酒色財氣重関，又躲不過生老病死苦処。倒不如雪月風花，得個偶紅倚翠。正是：
2 欲圖身外無穷樂，且尽生前有限時。
3 卻說隋煬帝大業元年，營昆仁宮，築西院，穷極華麗。宮樹秋冬凋落，則剪綵為花。月夜縱宮女遊幸，作清夜遊曲。至于欺娘姦妹，猶属荒淫。他又幸揚州看玩瓊花，想道陸地不便，令有司開汴河，駕龍舟，楊柳千尋，舳艫千里。死者相枕，百姓詐伪。群盜遍野，烽烟四起。唐高祖掃清宇內，乘著煬帝被宇文化及所弑，遂称大唐皇帝。于戊寅歲，改為建德元年。史官嘆惜隋亡，有詩八句為証：
4 紫微宮殿鎖烟霞，欲取蕪城作帝家。
玉玺不緣归日角，錦帆應是到天涯。
如今糞草無螢火，自古垂楊有暮鴉。
鏡破重圓陳后主，不宜再唱后庭花。
且說本傳中，一人家住荊州，姓武名曌，字行之，娶妻王氏，年過四十無子。王氏一日対丈夫道：「你我至今乏嗣，自古不孝有三，無後為大。莫若娶个偏房，好生一男半女，不知你心如何？」武行之微笑道：「既承美意，敢不如命。」遂去尋了個媒婆，娶得前村張木匠的女兒，到也俊俏。

Figure 2. Image-and-transcription view of the first page of a chapter of text (left), and the corresponding logical transcription view (right).[4]

Chinese Text Project Wiki

Wiki -> 洞玄靈寶自然九天生神章經解義 -> 洞玄靈寶自然九天生神章經解義卷一 -> Search for "靈寶君"

Search details:
Scope: 洞玄靈寶自然九天生神章經解義
Condition : Contains text 「靈寶君」
Total 10

《洞玄靈寶自然九天生神章經解義卷一》

6 靈寶君者，則洞玄之尊神，靈寶丈人則靈寶君之祖炁也。
7 此靈寶君為洞玄教主，諸義見前。
10 後至龍漢開圖，化生靈寶君。經一劫至赤明元年，出書度人，時號上清玄都玉京七寶紫微宮。
11 前日龍漢元年化生天寶君，此日龍漢開圖化生靈寶君者，蓋元年乃初劫也，謂之開圖乃化元顯發，文彩已露而靈寶君生於是時也。赤明乃第二劫名，屬南方天境。經一劫者，龍漢劫也。是靈寶君生於龍漢之中，而出書於赤明之初也。加度人二字者，蓋靈寶君實為傳教度人之祖。玄都玉京七寶紫微宮乃靈寶君所治之宮，亦藏經之所也。
34 天寶君應化於龍漢之初，當是時也，上無復色，下無復淵，溟涬大梵，寥廓無光。天寶以自然之章，開而闢之，則上下列位，天地開光也。靈寶君應化於龍漢開圖，乃為度人之祖，故以開九幽長夜之魂也。神寶君應化於赤明元年，寶為萬神之主，故云以制萬靈，制謂總御之也。太一即務猷妆，乃眾神之尊，故誦之則具足身中之神也。帝君即逢凌梵也。結形乃結炁成形也。九天即九天之帝也。以上蓋言誦章之功，利他之行也。夫誦與聞對，誦則施其惠，聞則受其惠。而學士居六對之間，乃自利者也，及其成功，則利他矣。雖隨其所用，而功各不同，然實一道也。

Chinese Text Project

Library -> 洞玄靈寶九天生神玉章經解義 -> 洞玄靈寶九天生神玉章經解義 17 /160

劫開九乃陽數經云積陽成天是也龍漢元年化生天寶君者龍漢乃始劫之名屬東方天境元年是初時之號亦非如今世之年數也所謂化生者乃有而無形無而有精非由形生故稱為化如洞神經云妙像無形應感有體真精妙炁化成姿容是也 按後同二 此乃無象之象非狀之狀惟道為身不屬空有矣出書時者即龍漢劫初之時出洞真十二部之書也且當是之時人物未兆何由出書殊不知是書之文是炁之所結也纔有是炁便具是書而炁之所以然者皆書之理也高上大有玉清宮乃天寶君之治所也

靈寶君者則洞玄之尊神靈寶丈人則靈寶君之祖炁也

[View] [Edit] [Quick edit] [Editing help]

Figure 3. Full-text search results can be displayed in context in a logical transcription view (left), as well as aligned directly together with the source image in an image and transcription view (right).

Creating transcriptions

The most fundamental types of material contained in the Chinese Text Project are digital facsimiles of pre-modern published works, and transcriptions of the textual content of these works. Although individual users can also upload both of these types of data individually, image data is more typically ingested in bulk through collaboration with university libraries and scanning centers, currently including the libraries of Harvard Yenching, Princeton University, and the Chinese University of Hong Kong, which have created high quality digital images of works in their collections. After ingestion, the next step in making these materials more useful to users is creation of approximate transcriptions from page images using OCR. Producing accurate OCR results for

historical materials is challenging due to a number of issues, including variation in handwriting and printing styles, varying degrees of contrast between text and paper, bleed-through from reverse sheets, complex and unusual layouts, and physical, water or insect damage to the materials themselves prior to digitization. In addition to these challenges, which are common to OCR of historical documents generally, OCR for premodern Chinese works faces additional difficulties in extracting training data due to the large number of distinct character types in the Chinese language. Most OCR techniques apply machine learning to infer from an image of a character which character type it is that the image represents, and these techniques require comprehensive training data in the form of clear and correctly labeled images in the same writing style for every possible character.[5] This is challenging for Chinese due to the large number of character types needed for useful OCR (on the order of 5000); unlike historical OCR of writing systems with much smaller character sets, it is not feasible to simply create this data manually. Instead, training data is extracted through an automated procedure (Sturgeon 2017a) which leverages knowledge about existing transcriptions of other texts to assemble clean, labeled character images extracted from historical works for every character to be recognized (Fig. 4). Together with image processing and language modeling tailored to pre-modern Chinese, this significantly reduces the error rate in comparison with off-the-shelf OCR software (Sturgeon 2018c).

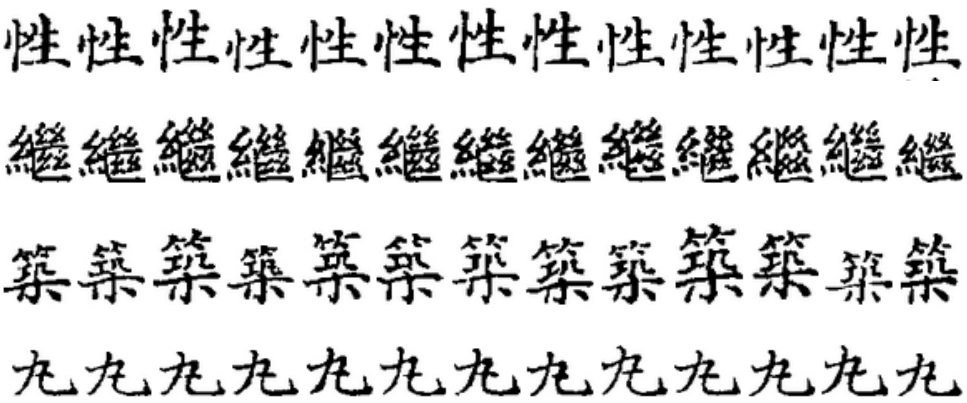


Figure 4. OCR training data is extracted automatically from handwritten and block-printed primary source texts.

Crowdsourced editing and curation

Once transcriptions of page images have been created, they are directly imported into the public database system. As initial transcriptions are created using OCR, they inevitably contain mistakes. Users of the system have the option to correct mistakes they identify, as well as to annotate texts in a number of ways. Two distinct editing interfaces are provided: a direct editor, which enables direct editing of the underlying XML representation, and a visual editor, which allowing simplified editing of page-level transcriptions, and which edits the same underlying content but does not require direct understanding or modification of XML. Regardless of the mechanism used to submit an edit, all edits are committed immediately to the live, public system. Edits are versioned, allowing visualization of changes between versions and simple reversion of a text to its state at any earlier point in time (Fig. 5). At present, the system receives on the order of 100 edits each day, generally

representing much larger numbers of corrections, as editors frequently choose to correct multiple errors in a single operation.

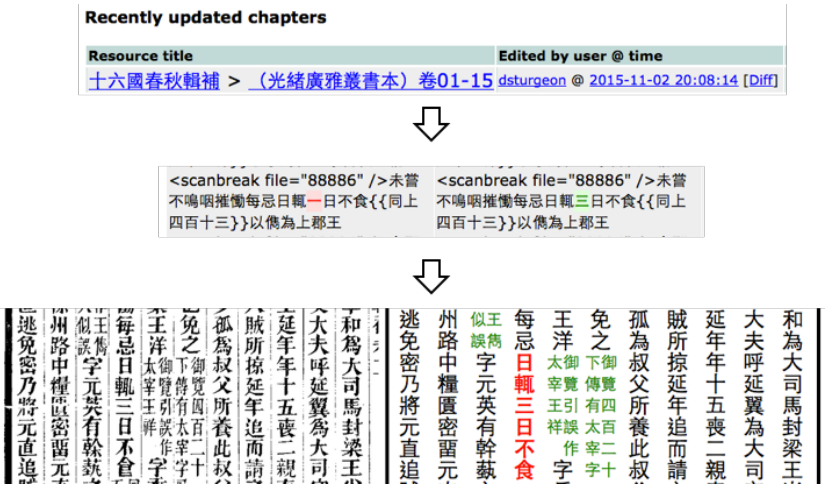


Figure 5. Visualization of versioning of a single crowdsourced edit. Arrows indicate interface workflow by following links from the edit log (top) to an automated comparison of the state immediately before and after the edit (middle) and then to the page and column corresponding to the edited text in image-and-transcription view (bottom).

Further visual editing tools supplement these mechanisms to enable crowdsourcing of more complex information. Illustrations are entered by the user drawing a rectangular box on the page image to indicate the location of the illustration, then filling in a simple form describing various aspects of it (Fig. 6). This results in an XML fragment describing the illustration, which can simply be inserted into the text at the appropriate location to represent it. This allows the illustration to be extracted from its context on the page and represented in the full-text transcription view as well as in the page-wise view. It also facilitates illustration search functionality, in which illustrations can be searched by caption across all materials contained in the system (Fig. 7). Caption indexing also allows comparison of illustrations aggregated according to caption where similar groups of captioned illustrations occur in multiple works, or in multiple editions of the same work; this can be used to quickly identify stylistic similarities and differences among works with overlapping groups of illustrations (Fig. 8). Future extensions to this functionality will also implement illustration search using image similarity comparison and ranking, allowing a user to search for illustrations similar to an arbitrary image file – greatly simplifying the task of identifying the source of an historical illustration of unknown provenance, as well as making it possible to identify visually similar illustrations occurring anywhere in the corpus.

A similar visual editing interface is used to enable the inputting of rare and variant characters which do not exist in Unicode. These characters are no longer in common use, but occur in many historical documents; additionally, as historical dictionaries fall within the scope of the project, character glyphs occur – and must be transcribed to give an adequate digital edition of the text – which may not be candidates for addition to Unicode at all (a common example being a dictionary entry which cites an otherwise unattested character form only to gloss it as in fact being

a mistake for another character). The visual editing interface for rare character input also uses metadata provided by the user to identify whether a given character is the same as any existing character known to the system, and if so, assigns a common identifier so that data about these characters can be aggregated, and text containing such characters searched.

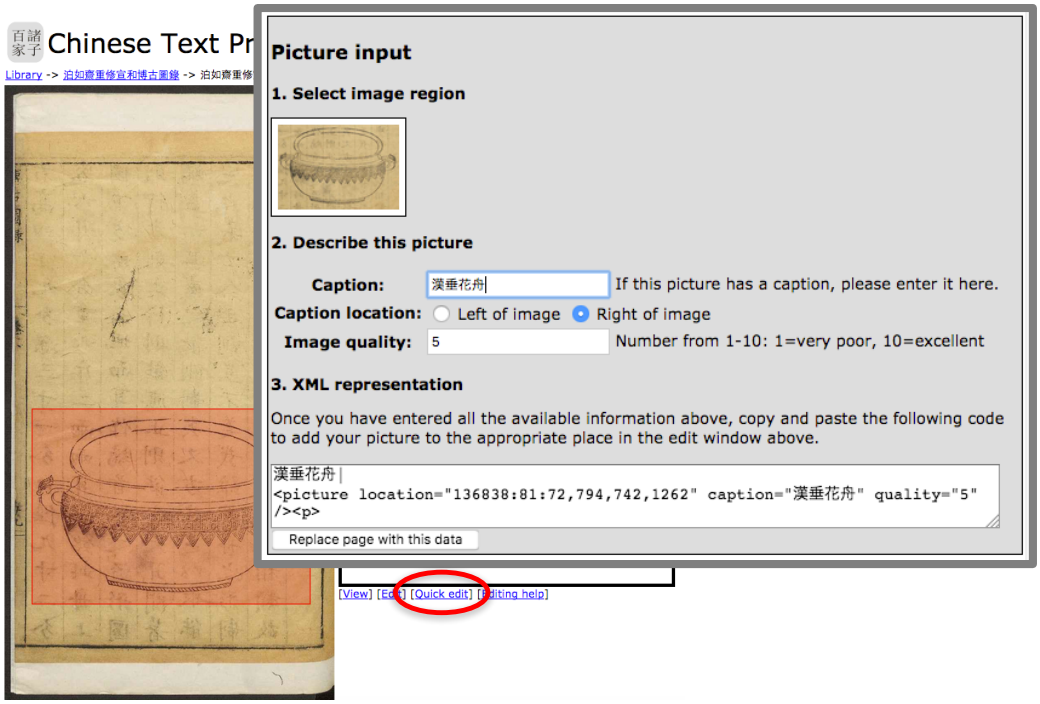


Figure 6. Identification and markup of illustrations within source materials are crowdsourced using purpose-designed visual editing tools which convert user input into XML.

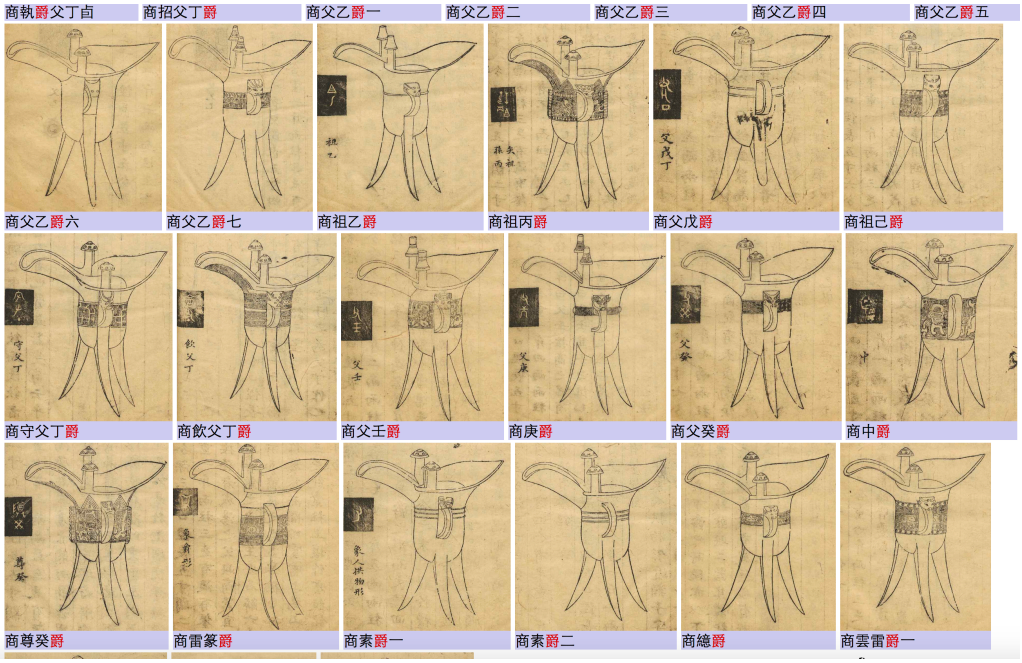


Figure 7. Illustration search: individual illustrations are extracted from (and linked to) the precise location at which they occur in source materials, and can be searched by caption.

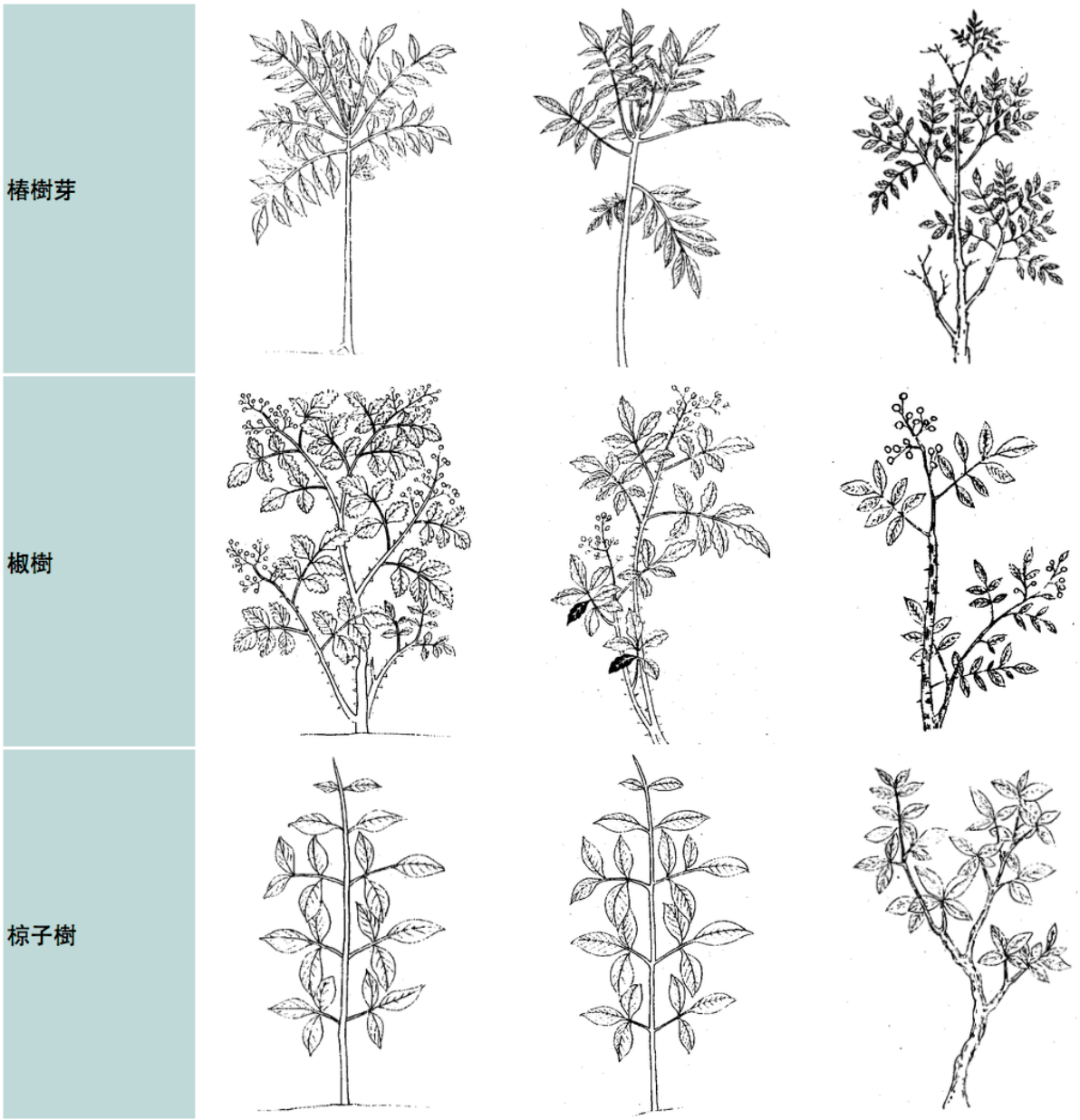


Figure 8. Tabular comparison of illustrations across multiple works: each column of images represents one individual work; each row represents a caption appearing in one or more of these works. This example highlights the clear stylistic similarities in the third row between the first and second work, which appear to have been copied either from one another or from a common source.

Application Programming Interface and Integration with External Systems

The platform and public web interface were originally released in 2005. Since then, alongside ongoing technical improvements to the platform itself, the volume of data accessible through the platform has grown by many orders of magnitude, from a few thousand characters of material in the earliest 2005 release, to over 5 billion characters in 2018. This rapid growth in content presents many opportunities for use cases not envisaged in the original design, particularly for text mining

use and statistical analysis of a sizeable and growing subset of all transmitted Chinese works. While some functionality supporting these uses has been built into the system itself – for example, dictionary pages programmatically extracting citations from historical dictionaries and aligned translations of example uses of terms from aligned translations of complete texts, and a searchable database of pre-computed text reuse relationships in the classical corpus (Sturgeon 2018a) – more open-ended access to textual data and metadata is also needed to facilitate other studies. Typical text mining tasks are expected to focus on subsets of the entire corpus, in part because the corpus itself intentionally incorporates duplication in the form of transcriptions of multiple historical editions of the same abstract work. Text mining tasks are expected to involve anything from some part of a single work, to many hundreds or even thousands of works.

In order to enable efficient access, a web API was created to deliver text and metadata for arbitrary subsets of the corpus in a machine-readable format. The goal of this was to facilitate two types of use: offline use, where data is downloaded programmatically for text mining purposes, and online use, in which independently developed web applications request and process data from the API in various ways – allowing for integration with other projects and the creation of "mashups" combining content from the Chinese Text Project with code and data from other sources. The API itself consists of a series of HTTP endpoints returning data in JSON format.[6] In order to facilitate use of the API in practice, two additional components are added: firstly, Uniform Resource Names (URNs), which are short identifiers uniquely identifying a textual object contained in the database, and secondly an XML-based plugin system, which provides a mechanism for users to extend the web interface itself. URNs representing textual objects are exposed directly from within the user interface, and can be passed to the API to request textual content and metadata relating to those objects (for example by being copied into a program or website capable of making API requests). Plugins provide a mechanism for users to have the web interface pass the URN associated with any textual object being viewed by the user within the web interface to an external web application via URL, enabling more intuitive connection between the web interface and other online tools capable of accessing content via the API.

Several additional design goals as well as properties of the materials themselves influenced the design of the JSON API. Firstly, the range of lengths of texts: some texts included in the corpus are very short – the *Daodejing*, for example, consists of only around 5000 characters – while others, such as the *Kangxi Zidian* dictionary at over 2 million characters long, are several orders of magnitude longer. At the same time, API responses cannot easily be precomputed, as the crowdsourcing system implies that textual content may change at any time. Lastly, the additional requirement of scalability – the ability for the system to handle large numbers of simultaneous requests from multiple concurrent users of the system – makes the prospect of assembling arbitrarily large textual objects in real time in response to API requests unpalatable, particularly given the possibility of maliciously constructed requests. Due to the popularity and high visibility of the public-facing system, this concern is not merely theoretical: to date, the web interface has, at various times, been the target of Distributed Denial of Service (DDoS) attacks involving hundreds of thousands of geographically distributed systems conducting simultaneous coordinated attacks. The API, like the web interface, must be as resilient as practical to such attacks, and a minimal requirement to achieving this is the avoidance of amplification effects, in which a small number of carefully chosen queries can cause disproportionately large amounts of work for the server responding to them.

In view of these requirements, API responses are designed first and foremost to be easily deliverable at scale. This is achieved by having API requests closely follow the internal representation of textual materials, in which all texts, large or small, are composed of smaller, more manageable units arranged in sequence. API requests for textual objects are therefore in general recursive: a request for a textual object will either directly return textual content, or instead return a list of URNs of other objects which a client must request and assemble in order to yield the complete text – no single API request will attempt to directly return a complete text composed of millions of characters. This 1:1 relationship with the underlying representation means that in general the work required to be performed by the API server scales linearly with the number of requests, avoiding amplification effects, and the number of possible distinct valid requests for textual data is limited by the total number of units in the underlying representation. This contrasts with more sophisticated text delivery APIs such as Canonical Text Services (Smith 2009, Tiepmar et. al. 2014), in which arbitrary spans of material up to and including complete texts can be requested in a single operation.

One consequence of this approach is an increase in API client complexity, as API clients must be capable of making follow-up requests as necessary according to API responses, and assembling full-text data accordingly. However, this additional complexity is easily mitigated on the client side, because the recursive request structure follows simple and predictable rules. Libraries for client languages and environments can use these rules to provide access to full-text materials with a choice of formats – current examples include open source libraries for Python and JavaScript which handle full-text recursive requests and provide simple wrappers for other API functions.[7]

Text Mining

A key anticipated application of the API is text mining of historical Chinese materials. The large size of the collection, together with the consistency of format guaranteed by the platform, API, and client libraries, make this body of material particularly attractive for such use. One challenge – and opportunity – in this field is that projects may be undertaken by researchers in many subject domains with widely varying technical backgrounds: while many computer scientists, Natural Language Processing (NLP) specialists, and corpus linguists will have the requisite skills to make use of the content and API functionality directly, there nevertheless remains an even larger potential audience of humanities scholars working with pre-modern Chinese materials, who may not have a technical background in programming, NLP, or text mining, but who have enormous potential to engage with digital techniques for exploring textual materials and apply them in focused ways in the context of their own research. This includes “DH curious” and casual audiences, but also students and scholars primarily based in traditional humanities departments who see the potential value of digital approaches to the study of textual materials.

In an attempt to engage this large potential audience of focused text miners and future digital scholars, text mining functionality has been added to the system (Sturgeon 2018b and 2019) in a way that does not require technical knowledge such as ability to use a programming language, by means of a separate system called Text Tools.[8] This functionality has been intentionally implemented exclusively using the API described above in order to ensure full separation between text mining code, and services implementing access to the data upon which text mining routines act. Instead of integrating text mining functionality into the core platform, Text Tools functions as an independent browser-based tool providing a user interface for basic text mining services, and

this tool is linked to the web interface using the plugin system and API in the same way as any other API application.

Text mining tools currently available in the tool range from the very simple, such as calculating term and n-gram frequencies, or performing regular expression search and replace – to significantly more sophisticated analyses such as identification and visualization of text reuse, and Principal Component Analysis (PCA) on user-specified vector data. Wherever possible, extensive use is made of interactivity, enabling exploration of the data and emphasizing the connection between numerical results and features of the texts which correspond to them. Many of the visualizations also make use of textual structures such as chapter divisions exposed consistently by the API to achieve intuitive ways of navigating the data. For example, text reuse can be visualized firstly as a heat-map layered upon the full-text content itself, but also as a heat-map matrix in which rows and columns correspond to chapters of text and cells represent reuse between chapters, or as a network graph in which nodes represent chapters and edges reuse relationships. In the latter two summary views, the cells and edges both link to the subset of the full-text heat-map visualization that corresponds to the two relevant chapters of text. PCA and cosine similarity comparisons also operate upon chapters as units of analysis, and present interactive visualizations in which numerical results are used to rank terms in decreasing order of their contribution to a transformed coordinate or cosine similarity score for a given result; these term lists themselves linking through to charts visualizing the relative frequency of that term throughout the corpus being analyzed.

In addition to interactivity, important benefits of the web-based interface and client-side processing approach adopted in this tool are ease and immediacy of use – nothing needs to be installed locally beyond a modern web browser – and scalability of service delivery. Additionally, the tool has been designed to be largely language agnostic and makes few assumptions about textual materials specific to the Chinese language. However, in order to perform useful text mining work with materials in other languages, some language-dependent processing will typically be required. For example, adequate tokenization rules for English differ from those for Russian, while tokenization for languages such as modern Chinese, written without spaces or other delimiters between words, require entirely different approaches to tokenization involving machine learning and models which might be prohibitively large to efficiently run in a web browser. Stemming, case normalization, and other types of pre-processing will be important for some languages, irrelevant for others, and in general differ in their details among those languages to which they apply. Many open source software packages and toolkits exist to perform these tasks for specific languages – however no single package combines all of these into one common system, and many different programming languages and dependencies are required by the various packages.

Instead of attempting to integrate code to handle language-specific pre-processing into the tool, an alternative distributed approach is used to address the problem. No language-specific routines are added to the tool itself, but a Text Transformation API is instead defined to allow transformations to be performed outside the tool.[9] This JSON-based web API is intended to be as simple and light-weight as practical, so that wrappers around existing tools can be easily created. An API server consists of a single endpoint offering two services: discovery – advertising what services are available, what languages they apply to, and how to access them; and transformation – taking input text and returning a modified text transformed according to the requested method. Thus the tool itself does not have any knowledge of any specific tokenization methods or other transformation services, but instead discovers at runtime which transformations are available,

either using the default API endpoint (https://txt.ctext.org/services.pl), or alternatively an endpoint input by the user.[10] This greatly simplifies the provision of a range of services based on different technical platforms, allows for daemon-like long-running services which can process requests quickly but take much longer to start up, and also makes it possible for a user to run services on the local system or elsewhere on a local network (as might be useful in a workshop or classroom setting) without modifying the user interface or workflow.

Conclusions and Future Work

This paper has demonstrated the feasibility of a dynamic approach to digital library design in which materials are simultaneously used and improved over time by their users. This approach has the advantage of giving access to a long tail of obscure material which might otherwise be at risk of lying neglected by full-text digital systems, awaiting its transcription and full correction through more traditional approaches.

The general framework introduced in this paper and implemented in the current online system has considerable room for further extension. Where possible, many future functions may be best implemented as independent, loosely connected systems, accessing textual materials as necessary via API, rather than being closely integrated into the library itself. Alongside Text Tools, the MARKUS textual markup platform is a good example of this, being developed entirely independently by a team using different technologies, and yet working seamlessly with textual data from the Chinese Text Project from a user perspective via API integration. Using the API, URNs, and a custom plugin, users can navigate textual materials within the Chinese Text Project web interface, then import these directly into MARKUS with a single click, and immediately have access to the specialized functionality of that platform. MARKUS users can also search and import textual materials from directly within the MARKUS platform using a dedicated search interface.

Future additions to the Chinese Text Project itself will likely focus on core functionality and content which benefits from maintaining close ties to the source materials themselves. This will include in particular more detailed semantic markup of the texts – easily implemented technically within the current framework using XML – to allow precise annotations of references to named individuals, places, and dates, together with enhancements to the crowdsourcing interface to facilitate this. Further enhancements to indexing of content, making use of this and other available data, will also be valuable improvements to the core platform.

Bibliography

De Weerdt, H., Chu, M. K. and Ho, H. (2016). Chinese Empires in Comparative Perspective: A Digital Approach*. Verge: Studies in Global Asias,* 2(2), pp. 58-69.

Holley, R. (2009). A success story – Australian Newspapers Digitisation Program. *Online Currents*, 23(6), pp. 283-295.

Neudecker, C. and Tzadok, A. (2010). User Collaboration for Improving Access to Historical Texts. *Liber Quarterly* 20 (1).

Newby, G. B. and Franks, C. Distributed Proofreading. (2003). *Proc. Joint Conference on Digital Libraries 2003*.

Smith, N. (2009). Citation in classical studies. *Digital Humanities Quarterly*, 3(1).

Sturgeon, D. (2017). Unsupervised Extraction of Training Data for Pre-Modern Chinese OCR. *Proc. FLAIRS-30.*
Sturgeon, D. (2018a). Unsupervised Identification of Text Reuse in Early Chinese Literature, *Digital Scholarship in the Humanities* 33(3), pp. 670-684.
Sturgeon, D. (2018b). Digital Approaches to Text Reuse in the Early Chinese Corpus, *Journal of Chinese Literature and Culture* 5(2).
Sturgeon, D. (2018c). Large-scale Optical Character Recognition of Pre-modern Chinese Texts, *International Journal of Buddhist Thought and Culture* 28(2).
Sturgeon, D. (2019 forthcoming). Accessible Text Mining with Text Tools and the Chinese Text Project, *Harvard Data Science Review*.
Tiepmar, K., Teichmann, C., Heyer, G., Berti, M. and Crane, G. (2014). A New Implementation for Canonical Text Services. *Proceedings of the 8th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH)*, pp. 1–8.

---

[1] Source: Google Analytics.
[2] Wikisource, by contrast, does not impose this restriction, and the majority of its transcriptions to not have corresponding page images.
[3] Summary of Formatting Guidelines. https://www.pgdp.net/c/faq/formatting_summary.pdf
[4] These pages can be accessed in the online system here:
https://ctext.org/library.pl?if=en&file=147636&page=13
https://ctext.org/wiki.pl?if=en&chapter=146021
[5] Similar comments apply to more recent techniques which require labeled text lines rather than labeled characters.
[6] https://ctext.org/tools/api
[7] https://pypi.python.org/pypi/ctext
[8] http://ctext.org/plugins/texttools/#help
[9] https://dsturgeon.net/tta/
[10] This default endpoint currently offers modern Chinese tokenization using Stanford CoreNLP, English tokenization using the Moses Statistical Machine Translation toolkit, Japanese tokenization using either MeCab or Kuromoji, part of speech tagging for English and modern Chinese using CoreNLP, as well as a handful of other textual transformations.