Mining dates from historical documents

Dana Mckay and Sally Jo Cunningham
Department of Computer Science
University of Waikato
Hamilton, New Zealand
{dmm9,sallyjo}@cs.waikato.ac.nz

Keywords: text-mining, digital libraries, metadata extraction.

Abstract
The essential quality of information in a digital library is accessibility. Full text search is not enough to ensure accessibility. Historical collections, for example, contain dates, and it would be useful to historians to be able to search by them. However, these dates occur anywhere within the text of historical documents, and to be searched they must be extracted from the documents and integrated into the collection index. Doing this manually is very expensive. Described here is the development of a system to do date extraction automatically, and use the resulting dates for searching. This system was implemented within the Greenstone framework used by the New Zealand Digital Library, and is based on carefully designed heuristics.

1. Introduction
Information in a digital library is supposed to be readily accessible; Vannevar Bush discussed the need for accessibility as long ago as 1945 [5], and it is still considered important today [7]. One of the advantages of a digital library over a conventional paper library is the ability to provide full text searching of its content (thus making it easier to locate documents and assess their relevance); digitally stored text can be indexed and searched with relative ease [10]. In some cases, however, full text search is lacking; the text does not tell the whole story on its own. One example where full text search is not enough is text used for historical research; full text search will often find names or places, even famous quotations, but the dates contained within these documents are another matter entirely. Consider as an example a historian who is interested in the fate of English peasants during 1066 invasion, they may search a digital library of historical documents using the search string ‘1066’, and this would indeed return all the documents containing the exact string ‘1066’, but not equally relevant documents containing, for example, “1061-69” or even “eleventh century”. Historians are interested in dates; they want to know what else was happening when a certain event occurred, or maybe what happened in the time leading up to it. Because full text search does not support historians as well as it might, a mechanism that will assist them in finding documents pertaining to a certain time period is a necessary part of making digital information accessible. Described in this paper is an attempt to create such a mechanism within the Greenstone digital library software, produced by the New Zealand Digital Library project (NZDL) [2].

Dates in historical documents can be considered metadata; that is, they tell us something useful about the nature of the document (in the case of historical document, dates tell us the time frame with which the document is concerned). Metadata is an important part of any library, whether it be titles, authors and call numbers stored in a card catalogue, or searchable information in an on-line index. Extracting metadata from source material, however, is difficult; this difficulty is compounded when the metadata in question is a latent part of the text (as dates are a latent part of historical texts). Manual extraction is time consuming, and it is a principle of the NZDL project that no manual processing of documents should be necessary to put them in a digital library. Automatic approaches for extracting metadata include compression-based techniques, syntactic analysis and heuristics. Compression-based techniques rely on metadata forming specific pattern when a document is compressed, and have been used for acronym extraction from computer science technical reports [12]. Syntactic analysis divides text up into its grammatical functions, and has been used in a phrase browsing system developed for the Food and Agriculture Organisation [9]. Heuristic analysis uses “rules of thumb” (usually provided by a human being) about what the desired metadata looks like to find the metadata within the text. Heuristics were used in a historical collection in London to extract names, geographical information,
and dates [6]. Automatically extracting the dates from historical documents was the main research goal and most difficult part of the work described here, by comparison the implementation of a search mechanism was simple.

The NZDL research group have collectively developed an open source digital library system called Greenstone. Using this system it is possible to create and maintain digital collections and serve them over the Web [11]. The collection building software and the collection serving software are separate entities, it is the collection building software that extracts metadata and the collection serving software that allows end-users to search and browse collections and read documents. The collection building software uses modules called ‘plugins’ to process different kinds of documents; there is a different plugin for each kind of document that can be incorporated into a Greenstone collection (for example HTML documents are processed by HTMLPlug and text documents by TextPlug). These plugins convert all documents to a Greenstone standard format known as Greenstone Mark-up Language (GML), and from there do any indexing and metadata extraction required. The person who owns the collection specifies which plugins are to be used (that is, what kinds of documents in their collection) and what metadata is to be extracted in a collection configuration file [4]. The collection building part of Greenstone is the part that was altered to allow for the extraction of dates from historical documents. The collection serving software includes a Web interface for collections (including searching and browsing interfaces), and CGI software to support these activities [4]. This part of the software was altered to allow historians to search for dates.

The Women’s History collection is one of the collections available within the NZDL [3], this collection was used as a test-bed collection for the research presented here. The collection is essentially a collation of relevant documents from various sources on the Web, harvested manually and built into a Greenstone collection. This collection contains documents covering a wide time span (from medieval times to the 1950s). Within the collection are many different kinds of documents, including primary documents, such as recipes, letters, poetry and newspaper articles; also included are documents about history like essays, biographies and dissertation. The wide variation within the collection, and its relatively small size (at the time of research 1075 documents) made it an ideal test-bed.

Section 2 discusses the issues involved in defining dates and mining them from text for use by historians. Section 3 describes the implementation of a date extractor, in section 4 the implementation of a searching mechanism is described. Finally there is a discussion of what was learned during this research, and a brief mention of work still to be done.

2. Defining the problem

From the outset of this project, it was accepted that the method for date extraction was to be heuristic evaluation of the text; this was largely due to the size of the project — of the techniques listed above, heuristic evaluation is the “easiest” (though perhaps not the cleanest). This meant that the first step of the project was to define dates, and identify potential problems with their extraction. To do this, a close manual inspection of the documents in the test-bed collection was carried out. The findings from this inspection are discussed in the following sections.

2.1. What constitutes a date

It was necessary to decide, even at this early stage in the project, what was going to constitute a date for extraction. Because no use could be foreseen for fine-grained dates (by and large granularity down to the day or even month level is quite rare, and so can not be useful for fine grained searches), nothing finer grained than a year was to be considered for extraction. This is not as straightforward as it might seem, though, because there are more kinds of dates than just the standard four-digit year. The collection contains some pre-Christian dates, year numbers ranging from one to four digits, ranges of dates (for example “1939-45”), and named centuries (for example twentieth century and 20th century); all of these represent time period information likely to be of interest to historians. As discussed below these patterns are sometimes difficult to distinguish from patterns formed by non-date numbers.

2.2. Text elements that look like dates

There are many naturally occurring numbers in text: monetary amounts, page numbers, street numbers in addresses, and count objects, to name just a few. The problem with these numbers is that they quite
often look enough like dates on the surface that the two are virtually indistinguishable; this is particularly the case for three digit dates, which are often interchangeable with page numbers and ordinal numbers. To avoid extracting these things that look like dates, or “date pretenders”, it is necessary to examine the context in which the numbers occur — sometimes the context can indicate that a number is definitely a date, and sometimes it will indicate with certainty that a number is a non-date (and sometimes, unfortunately, the context is ambiguous).

With so many different kinds of non-dates, there are many different textual environments that are strong indicators that a number contained within them is not a date. Some of these environments include preceding ‘p.’ or ‘pg.’, groups of three digits separated by commas (as in 3,425,689), and a preceding currency marker such as ‘$’. There are far fewer environmental indicators that a digit string is indeed a date, but the ones that do exist are very strong indicators — they work in 100% of cases in the testbed collection; these include a preceding or following ‘A.D.’, ‘BC’ or similar, or a preceding month name. These defining characteristics can all be used in the extraction process, and in particular it was decided that as many non-dates as possible would be pre-processed out of the text prior to date extraction.

Analysis of the source documents indicates that any four digit string less than the current year and not obviously marked as a non-date by some other factor (see Table 1) can safely be classified as a date. Two or one digit dates were always preceded or followed by an era marker (BC, CE, etc.); numeric strings of less than length three that lack an era marker can be safely discarded.

Three digit strings are more problematic. The vast majority of three digit strings in the test bed collection are not dates — they are more likely to be page numbers, street addresses, ordinals or microfilm references. It was determined through manual analysis of the source documents that there are a number of syntactic markers that guarantee a nearby string of three digits is indeed a date. These include prepositions such as ‘in’ and ‘since’ and again preceding time-related strings such as ‘the year’ or ‘spring of’.

### 2.3. Content-unrelated dates

There are many dates in the test-bed collection, which, while they are indeed dates, are not related to the content of the documents in which they are found. This is the case with dates contained in bibliographic material — they are dates, but may not be useful (documents about the 1500s that reference journal articles or similar written in the 1930s should not be returned in a search for information about the 1930s, for example). As such it is desirable to avoid extracting these dates as metadata — because they are not useful in assisting a user to meet their information needs.

Distinguishing between dates that are content related and those that are not is, however, no easy task because they appear to be — and indeed are — exactly the same format. Initially it was thought that a bibliography parser might be a viable option — but it was very quickly realised that parsing

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page</td>
<td>p, pg or pp (optionally followed by a ‘.’) or page, followed by a digit string</td>
<td>p. 104</td>
</tr>
<tr>
<td>Pages</td>
<td>p, pg or pp (optionally followed by a ‘.’) or pages, followed by a pair of digit strings separated by a ‘.’</td>
<td>pages 32-34 pp 32-4</td>
</tr>
<tr>
<td>Large number</td>
<td>A digit string of length 1-3, followed by groups of three digits preceded by commas</td>
<td>2,000</td>
</tr>
<tr>
<td>Colon</td>
<td>A colon followed immediately by a digit string</td>
<td>:37</td>
</tr>
<tr>
<td>Money</td>
<td>A &quot;$&quot; followed immediately by a large number or a digit string</td>
<td>$33,297 $2459</td>
</tr>
<tr>
<td>Microfilm</td>
<td>A string starting with the word ‘reel’ and ending with the first following ‘}’ or ‘.’</td>
<td>reel 8 #1098). reel 51 #9222</td>
</tr>
</tbody>
</table>

Table 1: Non-dates in text

Note that all text matches are case insensitive.
bibliographies is exceptionally difficult. For that reason an investigation was carried out into what kinds of things indicate that a date might be part of a bibliography, and it was discovered that there are certain kinds of punctuation patterns that indicate a date is likely to be spurious with high certainty (see Table 2). Furthermore, if the text before the punctuation is also examined (to see whether it is a month name and an ordinal number, for example August 21st or 21 August), even greater certainty about whether a date is content related or not can be can be achieved. Despite the fact that this solution is heuristically based, it is accurate, and easily implemented, and as such meets the needs of this project very well.

In sum, the problem of extracting content related dates from documents as metadata to be used for information retrieval is by no means as straightforward as it might at first seem. Upon closer examination the kinds of things that can represent dates are quite varied, and the digit strings that should not be extracted as dates (either because they are not dates, or because they are not content based) are at times indistinguishable from date strings. Investigation of the problem provided some insight into the kinds of tools that might be used in implementing a tool to do this extraction of metadata automatically.

3. Implementing the date extractor

The implementation of the date extractor was the most important part of the project — without it, all the information gathered about dates could not be used to assist users in their searching. The date extractor was implemented in Perl, as it is one of the major languages of Greenstone, and because the regular expression matching capabilities of Perl are ideally suited to the problem. The extractor can be used with any collection that can be built by the Greenstone software, and it is invoked at import time from a line added to the collection configuration file by the person building the collection. The extractor works in three phases, pre-processing, date extraction, and date storage.

3.1 The pre-processing phase

The first phase of the date extractor is the pre-processing phase. It uses a grammar based on all the things learned in the inspection of the test-bed (see section 2) to remove as much irrelevant digit material as possible. This is done in two steps, one that handles the non-date material (see section 2.2), and one that handles the content-unrelated material (see section 2.2). These tasks have been divided up to give the person building the collection some control over the material that is pre-processed out. If, for example, they think that their users may want bibliographic dates to be searchable, or if they know there is no bibliographic material in the documents that make up their collection, they can prevent the

<table>
<thead>
<tr>
<th>Context</th>
<th>% unrelated</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>,&lt;date&gt;</td>
<td>100</td>
<td>(Gies, 1978)</td>
</tr>
<tr>
<td>(&lt;date&gt;)</td>
<td>92.7</td>
<td>History n. s. 4 (1981): 3-34</td>
</tr>
<tr>
<td>. &lt;date&gt;</td>
<td>100</td>
<td>Ambassador Ltd. Canada: 1966.c</td>
</tr>
<tr>
<td>. &lt;date&gt;.</td>
<td>88.3</td>
<td>Libraries Press, New York, 1943.</td>
</tr>
<tr>
<td>; &lt;date&gt;.</td>
<td>92.3</td>
<td>Norton, New York; 1996.</td>
</tr>
<tr>
<td>., &lt;date&gt;.</td>
<td>96.1</td>
<td>Oxford University Press, 1940.</td>
</tr>
<tr>
<td>reprint of &lt;date&gt; edition</td>
<td>100</td>
<td>Reprint of 1942 edition.</td>
</tr>
<tr>
<td>winter, spring, summer, autumn OR fall &lt;date&gt;</td>
<td>100</td>
<td>Signs 5, 3 (Spring, 1980): 556-67.</td>
</tr>
<tr>
<td>bibliography</td>
<td>100</td>
<td>Bibliography: Power, Eileen. Medieval Women</td>
</tr>
<tr>
<td>work(s) cited: &lt;text up to end of file&gt;</td>
<td>100</td>
<td>Works Cited: Gies, Frances and Joseph.</td>
</tr>
<tr>
<td>references: &lt;text up to end of file&gt;</td>
<td>100</td>
<td>References: Carter, John Marshall</td>
</tr>
<tr>
<td>Site last edited/updated</td>
<td>100</td>
<td>site last updated 6 Jan 1999</td>
</tr>
</tbody>
</table>

Table 2: Content-unrelated dates

2 This percentage was determined using the unix `grep` tool.
3 Where <date> is a four digit number.
4 Greater accuracy can be gained by looking for preceding date material such as month names.
5 Note that all text matches are case insensitive.
6 Note that this and ‘References’ or ‘Works Cited’ alone on a line are sure indicators that bibliographic material is to follow and in no case did a content date exist beyond this point.
removal of non-content material by adding a line to the configuration file. This pre-processing takes place for two reasons; the first and more obvious being an attempt to remove anything that might accidentally be extracted as a date, the second being that this improves the processing speed of the extractor.

3.2 Extracting the dates

The second phase of the date extractor is the actual extraction phase. It is in this phase that the various kinds of dates are located and stored temporarily to await further processing. Some of the “dates” that are extracted represent more than one year, as in the case of ranges and century names (for example, thirteenth century represents all the years 1201-1300) and date ranges (see table 3). These cases are handled by expanding them to a list of all years covered by the century name or range, as this makes implementing the search process much more straightforward. Pre-Christian years are represented by prepending the string ‘bc’ to the year number.

In this phase the collection builder can specify a maximum year to be extracted, and a maximum century. If the collection owner knows that there is no content related material in their collection beyond a certain date they can specify that year in the collection configuration file as the maximum year. If there is no maximum year specified, the maximum year defaults to the present year, gained automatically from the system — this means that the maximum year will not outdated by the passing of time. The maximum year setting can be useful in discarding four digit strings that are not dates. The option to choose the maximum century to be extracted is included because as history approaches the present, centuries can become too coarsely grained to support effective searching, a search result set that includes all the documents containing the phrase "twentieth century" will have unacceptably low precision for someone looking for information about the 1930s. However, documents about more distant past may contain century names as their only reference to time, and it would be detrimental to information seekers needs to ignore centuries completely in this case. The collection owner may again specify the maximum century in the configuration file, but in this case the default is not to extract any centuries (because users find a search that returns large numbers of irrelevant documents as frustrating as those that return none [8]).

3.3 Storing the dates

The association of these dates with their documents as metadata is relatively straightforward — they are added in Dublin Core format to the GML document associated with the unprocessed material. It was unusually difficult, however, to determine the best element in the Dublin Core set to label the dates. The Dublin Core specification allows multiple entries of the same element, and additions of elements to its core set. Using an added element for the entire date list was considered but it was seen as more sensible to use multiple entries of one of the core elements, so as to be compatible with other systems using Dublin Core. The question then was which element to use — Dublin core has a date element, but it is specified as the date of creation of a document [1], and this is not what the extracted dates represent. The coverage element of Dublin Core, however, may pertain to the temporal qualities of the content of a document [1], and may contain single dates or date ranges — and as such extracted dates are stored as coverage elements containing single dates.

The date extractor is, in effect, a translation of everything learned about what dates look like and what date-like things are useful to information seekers into some rules a computer can understand. This is the most crucial part of developing a search by date feature for a system using automatic metadata extraction, because it creates the dates over which the collection users search. The extractor is designed in three phases with careful consideration given to each phase, so that it best represents the information needs of potential users of historical collections. Where these needs are not completely clear, control has been given to the collection owner, who may know more about the source documents of a collection and its users, and therefore be able to fine-tune the extractor.

4. The search software

There were two parts to implementing a search-by date feature within the Greenstone framework, the first was creating an interface that allowed users to enter dates to the search software, and the second was creating the underlying search mechanism. The requirements of the search software and interface
included allowing users to enter a single date or a range of dates, and allowing ordinary “dateless” searches to take place or incorporating dates or date ranges into a search as required.

4.1 The search interface

A search interface in Greenstone has three elements, the interface layout, the language in which the interface is presented, and the help files associated with the interface. The first step in designing search by date interface was laying it out — the layout was dictated largely by the standard Greenstone interface, but additions to accommodate the entering of dates to the interface had to be made. The two date entry boxes (see Figure 1), and the drop down boxes to select the era in which a date is to occur (radio buttons were considered, but this was not consistent with the rest of Greenstone’s interface) are the only additions made to this part of the interface. The drop-down boxes, in keeping with modern sensibilities, use the culturally sensitive ‘C.E.’ and ‘B.C.E.’ rather than ‘B.C.’ and ‘A.D.’. The help files developed were a simple extension of the standard Greenstone help.

4.2 Implementing date search

The date search feature was required to have all the functionality of a standard Greenstone interface while incorporating dates. This meant that a user had to be able to search for text only, a single date, any document containing a date within a given range and any combination of text and dates. There were many possible ways to implement these requirements, including post processing text search results to remove documents not meeting text requirements, and altering the query string.

Post-processing result sets was rejected as a method for implementing date search because within the Greenstone system it proved to be somewhat inaccurate (at the expense of some recall). Thus altering the query string was the method chosen for implementing a date search. This alteration occurred at a level underlying the interface, so as not to confuse the user by altering the search string they entered on-screen, and made careful use of Boolean logic so as not to alter any textual part of the query in any way.

The search software added to Greenstone is sufficiently unobtrusive that searching a date-enhanced collection without any date constraints is unchanged at both the interface and functionality.
levels. However, the changes to the standard Greenstone search are sufficient to provide a simple, obvious and accurate date search.

5. Conclusions

Different kinds of digital collections require different access methods to be as useful as possible to their users. Full-text searching goes a long way towards providing good access to many kinds of collection but in some cases it is not enough. One such case is dates in historical documents; for historians to be able to effectively locate and use dates a new search mechanism was needed. To provide this mechanism it was first necessary to define dates, then to create an engine to extract them, and finally to create software for searching them.

By far the most difficult part of creating a search-by-date mechanism is creating software to extract dates from the documents. However, it is the accuracy of this part of the process that is vital to historians; it ensures that only useful dates are searchable, and that as many of the useful dates in a collection as possible are available for searching. If date extraction is done poorly the resulting search-by-date feature will be nearly useless.

6. Future work

Planned for the future is an evaluation, to be carried out by historians, to verify that this software is extracting useful dates. It is believed that how well the extractor performs in this regard is heavily dependent on the knowledge of the collection owner about their collection and their potential users.

Another piece of work for the future is the implementation of an era search scheme. The idea behind this is that the owner of a collection could specify eras from history (be they Ming and Tang or Renaissance and Cubist), and their start and end points in the Julian calendar in the configuration file. Rather than having to specify dates to restrict their search by, a user could restrict their search to a result set from a given era, chosen from a menu on the user interface.

7. References


Acknowledgements

Without the hard work of Annette and David Bainbridge in collating and creating the Women's History collection, none of this work would have been possible — many thanks.

Warm thanks also to Stuart Yeates for his encouragement, support and incisive mind throughout the project.
