Open Boek: a system for the extraction of numeric data from archeological reports

H. Paijmans and S. Wubben
Tilburg University
RACM Amersfoort
paai@uvt.nl, S.Wubben@uvt.nl

June 20, 2007

Abstract

This paper describes the current state of the Open Boek information retrieval system for natural language archaeological papers and reports in the Dutch language. The system focuses on the recognition of phrases that contain chronological and geographical references. Memory Based Learning is applied to assign the correct classes to such phrases; after that indexes are created for later retrieval and optionally tags are inserted to perform appropriate actions such as linking to Google Maps. In this paper refinements to the original modus operandi and new problems and solutions are described.

1 Introduction

In archaeology, as in all other cultural heritage domains, most knowledge is stored in articles and books, i.e. natural language text. Retrieval of facts and information from such texts is notoriously difficult, unless we confine ourselves to keywords. This means that we can search (on Google, for instance) for the string 'middle ages', and retrieve all documents in which those words occur. But Google does not know that *'eleventh century'*, *'1300-1412'*, *'XIIth century'* and countless other chronological expressions would all be relevant for somebody who is interested in the middle ages. In our system, Open Boek, we address these and related problems.

Elsewhere [Paijmans and Wubben2007] we described the principles and constraints that governed our approach to the problem of information retrieval in dutch archaeological texts. Essentially we do not try to create or use a general ontology or other such 'grand design' such as CIDOC/CRM, to interpret the contents of a text, but we try to solve the recognition of each semantic class on its own merits, strive for a satisfactory performance and then go on to the next class.

This is not to mean that ontologies such as CIDOC/CRM are not useful for structuring data, to impose a common standard or even as a tool for extracting data from NL (Natural Language) documents; see also [Généreux and Niccolucci2006]. But at this stage we feel that the community is better served with immediate solutions, that in their turn may suggest ways and means for more involved approaches and ontologies. And of course the indexes that we generate can at any time be translated to XML or included in a database.

As a case, let us consider an institution such as the RACM[1], where a large number of papers and reports about archaeological excavations, site surveys and similar documents are stored digitally. Access to the information in the reports is by a collection of separate databases in which relevant attributes of the documents are entered by human operators, by straightforward scanning for keywords, or, sometimes, by a rudimentary keyword index. Although there is traditionally much activity in the archaeological world in the field of typologies and controlled dictionaries, and although there is an urgent need for so-called 'reference collections' that support such typologies [Lange2004], there is no agreement on how to apply such typologies to information retrieval. This is typical for the archaeological scene; there exist projects to create a more involved XML markup for documents, based on CIDOC/CRM, e.g, by J. Holmen and his collaborators, [Holmen et al.2003], but here no automatic extraction from instances in the text into the tags is envisaged, and the current status of the project is not clear.

The needs of the archaeologist are concisely ex-

[1]Rijksdienst voor Archeologie, Cultuurlandschap en Monumenten, the central authority that collects data and monitors archeological activity in the Netherlands

pressed as "What, when, where" and our approach is to begin with a chevalier solution for the *what*; i.e. weighted keyword access in the Vector Space Model [Salton and McGill1983]. In the technical sense this is not a major problem, as there are many good keyword based retrieval systems for plain text based on the VSM. Then we proceed to extract the semantic content needed for the *when* and *where* in stages. For this, we are constructing an information retrieval system, **Open Boek**, that automatically extracts, translates and indexes such attributes from the text. At the moment, such data are used exclusively for retrieval, but in the long run the findings will be used for more involved operations, notably the identification of objects and collection of the data that are relevant for those objects.

But after this stage is completed, we will return to the *what* with a vengeance, when more involved applications of NER (Named Entity Recognition) will be applied to recognize objects and information related to those objects. In this paper we present our progress in the recognition and interpretation of chronological and geographic data.

2 The documents

The design of **Open Boek** [2] hinges on a few requirements. As we already mentioned, the system should be both immediately usable from the beginning of the project, and remain open for additions and changes. Therefore we adopted a tight modular approach, where the modules communicate by ASCII files, using Unix text tools where possible. This may have had some impact on performance, but it certainly makes it easier to inspect intermediary results. Also, it invites experimenting with different modules that have similar functionality. Of course, our own programs, and the programs on which Open Boek depends, are all Open Source, and where possible licensed under the GPL.

2.1 The original formats

An important constraint is the format of the original documents. We had access to a large collection of thousands of reports in all kinds of formats. They can largely be classified in three groups, which we will discuss below.

1. By far the largest portion (about two thousand reports of approx. fifty pages each) were originally typed on paper, and later scanned, OCRred and stored as PDF. In such files, the 'image' of every page was paired by an 'invisible' ASCII text that however could be easily extracted and indexed. The problem here was the display of the retrieved pages. The original pdf-images of course contain all sorts of pictures, tables and drawings, but we did not address the technical problem of highlighting keywords or the addition of links in that 'visual' pdf-representation. Instead we converted the contents to HTML for that purpose.

2. Another large portion of the files was already written using a word-processor and stored as PDF. Unlike the first group, there was no OCR step necessary, and the quality of the text was better. Such files translated relatively easy in HTML, combining highlighting, links and images.

3. A third group of documents consisted of hundreds of reports written by individual archaeological bureaus. These were stored on as many CDs and almost always produced with Microsoft software. Without a doubt every CD contains a highly artistic multimedia feast with sounds, movies and everything, but it was absolutely impossible to extract the original reports without a time-consuming process of analyzing the contents by hand, defeating the purpose of *automated* indexing and retrieval. But even if the 'central' document could be identified, Microsoft's OLE framework essentially prevents precise extraction of the relevant data, at least with the tools that we used. Also, using those tools, we often find pictures or phrases in the Word documents that obviously are not meant for publication and are included without the author being aware of it.

So we limited ourselves to the PDF HTML format, translating every page into HTML (therefore HTML-files can also be used as input). Although the HTML-ized pages were not always rendered correctly (especially not in the first group), we could easily highlight 'interesting' phrases and add links to e.g. Google Maps. Of course the original pdf-image of the page can be displayed at any moment, but at the cost of the highlighting and the links.

3 Open Boek: processing

As a first step, the pdfs are converted to individual HTML-pages and separate images. Then, the text proper was extracted from the HTML. One typical database, scanned from paper and OCR-red, consisted of 750 pdf documents (1.7 Gb) extracting to 50 MB ASCII text, contained in 30.000 pages and as

[2]The programs and documentation can be found online at http://www.referentiecollectie.nl/Openboek

many images. The extraction of the HTML from the PDF files is done by *pdftohtml* [3], from which the final ASCII text is produced. For normal text this poses no particular problems, but 'landscape pages' generally become casualties. Also, text in columns loses its coherency, which is a problem when interpretation depends on text windows, as is often the case in MBL. We store the HTML tags and the word tokens in separate files, that are combined only when the page has to be rendered in a browser. This 'standoff' notation makes it easy to add tags in a later stage, e.g. to mark chronological or geographical content.

3.1 Keyword indexing

The documents are indexed at the document level and at the page level. The purpose of this twofold indexing is that combinations of keywords can be applied at both levels. We used the venerable SMART program, developed between 1965 and 1995 by Salton [4]. SMART is an implementation of the Vector Space Model [Salton and McGill1983], which essentially retrieves documents on keyword combinations and, most importantly, ranks them according to some measure of relevance.

The SMART program offers several distinct weighting methods for individual words and we are still debating which is the best for this particular purpose. In any case, it serves as a fast and reliable indexing and retrieval engine and so can be the basis for a very usable document retrieval system. The creation of the keyword indexes for a database of this size is typically a matter of a few minutes on a modern PC running SuSE Linux.

3.2 Indexing of numeric and geographic features

If SMART takes care of the *what*, the problems remain of the *when* and *where*.

The indexing of geographical features is as yet in its experimental stage. The purpose of this indexing is twofold: first to be able to search for such locations in terms as '*...within a circle of ten kilometers round Amersfoort...*' or '*...inside the countyborders...*'(see section 3.4). The second is disambiguation: which county is meant in the last example? A monument could, e.g. be called "Loevenstein Castle" and exist in a database somewhere with exact location, coordinates and so on. In the text of a document however, it could be referred to as 'the castle' or even 'the building', but our system should still be able to identify the castle from context and add precise information.

3.3 Chronological indexing

At this moment, we have implemented two modules to do the chronological indexing: one is based on MBL (Memory Based Learning) and the other is a plain rule-based script. In both cases they create an index with all dates occurring in the document and store them as periods in the index-file. The location in the document is also tagged, although this is not essential. The final result is that the system "knows" that a particular year or period lies within the 'Middle Ages', or in the twelfth century, or in the XII-th century or whatever phrase is used in the document, and so is able to present texts, relevant for that particular date.

The chronological indexing proceeds in three steps. First, the candidates for classification are collected by a numeric pre parser. This pre parser recognizes not only items like *2* and *100*, but also the written, the Arabic and the roman cardinals and ordinals like the dutch equivalents of *two*, *second*, *2nd* or *2-nd*, *VI*, *VI-th* etcetera). In this phase, also a list with names ('middle ages', 'iron age', 'roman period') is consulted and the corresponding phrases are also flagged as chronological phrases.

In preliminary experiments, we had both a MBL and a rule-based classifier. Although the rule-based classifier was much faster, the performance in terms of precision and recall was far below that of the MBL classifier, and we discontinued experiments with it.

The third and last phase is the normalization and the creation of the index proper. This includes assignment to BC or AD, and the decision whether the expression contains a single year, or is a period. 'between 1200 and 1300' obviously is a period, but so is 'third century'. More complicated are expressions as 'between the first century BC and the year 500', and we are still working to perfect our scripts to parse all possible combinations.

3.3.1 Memory Based Learning

To automatically detect potential timespans we use a form of Machine Learning. Memory Based Learning is a method of storing a set of training examples in the 'memory' and for each instance that the system encounters this set is checked. Using statistical methods the most likely candidate is then selected and the according tag assigned.

For the Memory Based Learning we used TiMBL 5.1 [Daelemans et al.2004] [5], a decision-tree-based

[3]http://pdftohtml.sourceforge.net

[4]For more information on SMART and a tutorial see: [Paijmans1999]

[5]Available from http://ilk.uvt.nl

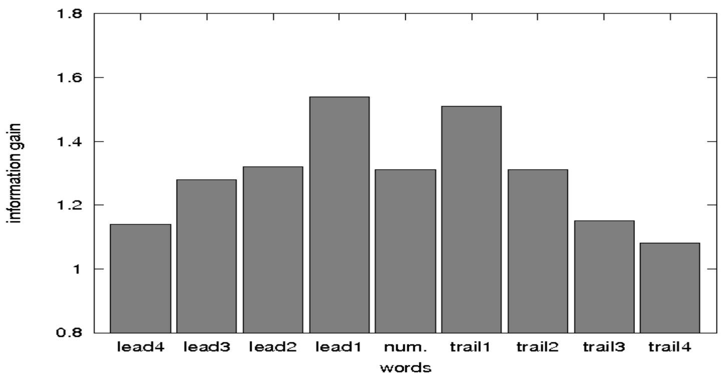

Figure 1: Information gain of all nine features

implementation of k-nearest neigbour classification (KNN). KNN classification is a method of classifying objects based on the closest training examples mapped in the feature space. TiMBL uses indexes in the instance memory extensively and therefore can handle discrete data and large numbers of various examples well.

The settings that proved most efficient for a development set consisting of 25,507 random instances were:

- *IB1 classification algorithm*: a simple instance-based learning algorithm. It simply stores all example instances and finds the closest one. IB1 uses a similarity function where the instances are described by n attributes. The algorithm is quite similar to the k-nearest neigbour algorithm [Aha et al.1990].
- *Modified value difference metric*: a method of looking at co-occurence of values with target classes to determine how similar the values of a feature are, using exemplar weights [Cost and Salzberg1993].
- *Information Gain feature weighting*: measures the discrimination value , i.e. how much each isolated feature contributes to the correct classification.
- *Five nearest neigbours used for extrapolation*: five nearest neigbours are used for classifying objects based on closest training examples in the feature space.
- *Exponential Decay weighting with parameters* a = 1.1 *and* b = 1.1: gives exponentially less weight to more distant neigbours for classification.

These settings were used to perform a tenfold-cross validation test on the remaining data (22,563 instances). The numeric classes are based on CIDOC/CRM.

With a total of 94% of the instances classified correctly, the MBL-component performs well and implementing it is therefore acceptable. Classes that don't perform very well are, not surprisingly, generally the smallest classes. When we confine ourselves to only three classes ((chronology, coordinates and other) the performance climbs to 98%.

As demonstrated in earlier research [Buchholz and van den Bosch2000], figure 1 shows us that the closer the word is to the focus, the higher its information gain becomes. This means that words closer to the focus (in our case a numeric) are more important for the classification of that numeric. When at equal distance, words in front have a slightly higher information gain value than trailing words. Something that makes our situation interesting, is the fact that the information gain of the numeric itself is actually *lower* than the information gain of its direct neigbours. This means that the numeric itself contributes less to its classification than the words directly in front or in the back of it. These results stress the ambiguity of numerics in archaeological texts and the need to use context to disambiguate the numerics before the numerical information therein can be made explicit.

3.4 Indexing geographical features

Now that we have taken care of the 'what' and the 'when', it is time to address the next challenge: the 'where'. We want to give the user the opportunity to narrow his query down to a specific geographic area. The user may for example want to search for all fibulae[6] found near Zwammerdam. [7]

'Near Zwammerdam' means the system should return all reports dealing with fibulae found in

[6]A fibula is a brooch used by the Romans to fasten clothing
[7]In Roman times, Zwammerdam was called Nigrum Pullum and produced Roman cargoships for transportation on the Rhine

Zwammerdam, but of course also in the places that are located around Zwammerdam. For this, geographical knowledge is required. We used a short-form gazetteer containing all placenames in the Netherlands, accompanied by the latitude and longitude coordinates. We recalculated these coordinates into WGS 84 (World Geodetic System) coordinates. WGS 84 defines a reference frame for the earth, for use in geodesy and navigation. We decided to use this system as it is widely used, for example by Google Maps, and it is relatively easy to do calculations on the coordinates.

If the user wants to search for an area around Zwammerdam with a radius of 5 kilometers, we need to calculate the distance from Zwammerdam to all other places and select only those that are less than 5 kilometers away. For this calculation the Haversine formula was used [Sinnott1984]. This formula is an equation that gives great-circle distances between two points on a sphere from their longitudes and latitudes.

Using this equation we can find all placenames that have a distance smaller than 5 km from Zwammerdam. This yields the places Aarlanderveen (4.46 km), Bodegraven (2.89 km), Korteraar (4.82 km) and of course Zwammerdam (0 km). A query can now be done to search for all reports about fibulae found in these places.

We could simply add the placenames found as keywords in our search, but this poses another problem. In some cases, placenames are not appropriate for the query. This is for example the case when a placename is part of a publication, or when a person bears a name that is similar to a placename. In our search for fibulae near Zwammerdam, we do not want to find any reports dealing with fibulae that refer to publications that were published in Zwammerdam, or persons that are called 'Van Zwammerdam'. A combination of heuristics and Memory Based Learning can help us here. Roughly 90 percent of all potential placenames in the archaeological reports we have access to is actually a relevant placename. This leaves ten percent of irrelevant keywords. Using Memory Based Learning, trained on 'good' and 'bad' placenames this can be narrowed down to approximately six percent.

Finally we use some simple heuristics to recognize pages with literature references, and omit these from the placename index.

A side project for KICH focuses on identifying monuments in reports. A monument database is used to geolocate monuments in these reports. Because we already detected references to coordinates that exist in the text, all documented locations of excavations and monuments can be searched as well, giving the user the option to search for monuments and excavations in a specific area and viewing these locations in for example Google Maps.

3.5 Retrieval

The keyword retrieval is based on the Vector Space Model, but offers three weighting methods for the results of the search: boolean, frequency-based and atc-weighted ('atc' weighting is a $tf.idf$ weight that takes in account the length of a page). The queries are resolved by SMART itself.

More interesting is the processing of a chronological query. As we have seen, the various indexing modules (except SMART) create simple ASCII files where the information about chronology and other classes is stored explicitly (see table **??**. In the example we see the file, the page number, the beginning of a time period and the end of that period.

At retrieval time the user enters a simple expression that is either a single year (500), a period (500-1500), the name of a period ('middle ages' or even '200BC - middle ages'), that are compared to the periods in the time index. He can also indicate whether his query should completely encompass the years and periods in the file, or that overlap at one or the other end is allowed. In the first case, the query '500-1500' will retrieve all pages on which references to years and periods within the middle ages (including the middle ages itself) occur. In the second case, also pages that refer to periods beginning before the middle ages, but ending within 500-1500, or conversely, periods that begin in the middle ages, but continue after 1500.

We will now give a small demonstration of the information that can be gleaned from the reports if and when the chronological information is made explicit. From the current system it is already relatively easy to map e.g, chronological references as in fig. 2, where references to the all years and periods within 1000 BC-2000 AD are shown in a database with reports on dutch archaeological sites. A few interesting features are visible from left to right. First, the plateau caused by frequent references to the 'Iron age' defined as between 800 BC and 50 BC. Then a small surge that starts sharply with the year 0 AD and rapidly falls off. This surge is caused by the incorrect classification of e.g. page-numbers or paragraph numbers as years. The other spikes in the graph are caused by the human tendency to gravitate to 'round' numbers. This is very visible in the years 50 BC, 500 AD and 1000 AD and to a lesser degree every 100 years. The middle ages in themselves are visible as another plateau in the graph; caused by the frequent use of the term 'middle ages'. The activity then is lower for the next few hundred years, but rises a bit towards the year 2000 as a result of bibliographic references, that of course

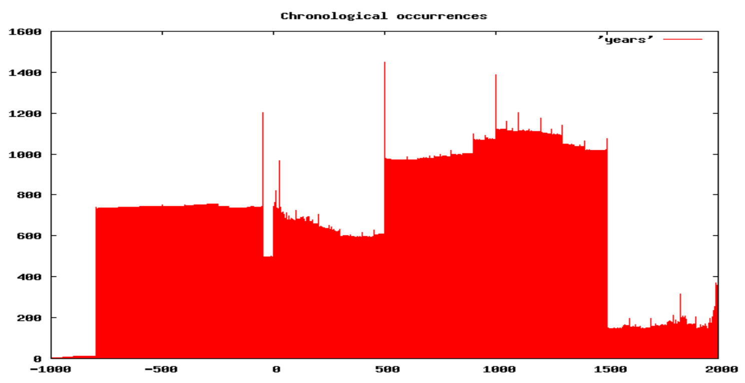

Figure 2: References to years between 1000 BC and 2000 AD in RACM reports

also include the year of publication. We have something to say about that in section 4 below.

4 Open Boek: the future

We mentioned already that the recognition of numeric values and geographical references are only the first steps in the creation of the Open Boek system, and that the project has two very distinct goals. The first is to build a text retrieval system, that incorporates modern techniques for the interpretation of certain semantic classes, such as chronology or geography. To complete this stage, we have the following tasks before us:

- An Open Source stemmer for Dutch should be selected and implemented, to reduce the number of keywords.
- If and when performance becomes a problem, indexes and other data should be stored in a SQL-database, but that entails drastic redesigning of the system. Currently most index files are plain ASCII, and are processed by the standard Unix text utilities.
- The problem of bibliographical references mentioned above is not yet satisfactory solved. A better solution would be to build a system that recognizes such 'fremdkörper' and places markers around them.

Still more interesting (and more difficult) is the final task set before us: to carry the interpretation of NL text to the point that we can identify phrases, passages and images that refer to (archaeological) objects mentioned in the text. We will describe our ideas and approach to that problem in a different paper.

[8]Continuous Access to Cultural Heritage
[9]Reading Images in the Cultural Heritage
[10]Mining for Information in Texts from the Cultural Heritage
[11]KennisInfrastructuur CultuurHistorie

Acknowledgements

This work was supported by NWO/CATCH under grant 640.002.401.

The system, Open Boek, is a 'use case' for the CATCH project [8], more in particular for RICH [9] and MITCH [10], but we also are supported by other research groups in the project, such as KICH [11] and the National Reference collection(NRc).

No Microsoft software was used for the experiments mentioned in the paper or for the preparation of the paper itself.

References

[Aha et al.1990] David W. Aha, Dennis Kibler, and Marc K. Albert. 1990. Instance-based learning algorithms. *Machine Learning*, 7:37–66.

[Buchholz and van den Bosch2000] S. Buchholz and A. van den Bosch. 2000. Integrating seed names and n-grams for a named entity list and classifier. In Dybkjaer [Dybkjaer2000], pages 1215–1221.

[Cost and Salzberg1993] Scott Cost and Steven Salzberg. 1993. A weighted nearest neighbor algorithm for learning with symbolic features. *Mach. Learn.*, 10(1):57–78.

[Daelemans et al.2004] Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. 2004. Timbl: Tilburg memory based learner, version 5.1, reference guide. ilk technical report 04-02. Technical report, Tilburg University.

[Dybkjaer2000] L. Dybkjaer, editor. 2000. *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC 2000)*.

[Généreux and Niccolucci2006] Michel Généreux and Franco Niccolucci. 2006. Extraction and mapping of cidoc-crm encodings from texts and other digital formats. In M. Ioannides, D. Arnold, F. Niccolucci, and K. Mania, editors, *The 7th International Symposium on Virtual Reality, Archeology and Cultural Heritage (VAST 2006), 30 Oct - 4 Nov, Nicosia, Cyprus.*, pages 74–79.

[Harman and Voorhees2005] D. Harman and E. Voorhees, editors. 2005. *The Fourteenth Text REtrieval Conference (TREC-14)*. National Institute of Standards and Technology.

[Holmen et al.2003] J. Holmen, C-H. Ore, and O. Eide. 2003. Documenting two histories at once: digging into archaeology. In *CAA 2003 - Computer Applications and Quantitative Methods in Archaeology*. Bar International Series 1127 2004.

[Lange2004] A.G. Lange, editor. 2004. *Reference collections, foundation for future archaeology*. Rijksdienst voor Oudheidkundig bodemonderzoek, Amersfoort, may.

[Paijmans and Wubben2007] J.J. Paijmans and S. Wubben. 2007. Memory based learning and the interpretation of numbers in archaeological reports. In M-F Moens, T. Tuytelaars, and A.P. de Vries, editors, *Proceedings of the 7th Dutch-Belgian Information Retrieval Workshop*, pages 51–56.

[Paijmans1999] J. J. Paijmans. 1999. *Explorations in the Document Vector Model of Information Retrieval*. Ph.D. thesis, Katholieke Universiteit Brabant.

[Salton and McGill1983] G. Salton and M. J. McGill. 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill New York [etc.] - 448 pp.

[Sinnott1984] R.W. Sinnott. 1984. Virtues of the haversine. *Sky and Telescope*, 68(2):159.