_index

Open Boek: technical report and manual, version 1.0

Hans Paijmans, Sander Wubben

Abstract

This report provides a description of the Open Boek intelligent retrieval system version 1, and of its care and feeding. It combines the user manual and the administration guide. Finally, it provides detailed descriptions of the scripts and fileformats.

1 Introduction

Open Boek is the 'use case' of the two CATCH projects RICH and MITCH. It aims ultimately at the extraction and combination of textual and visual data from written documents so that databases of images and corresponding data can be created from reports in natural language. As a first stage, we implemented a system that can recognize the semantics of numeric data for, e.g. chronological search and retrieval[4, 5].

This report is a description of this first stage of Open Boek. It provides information for the end user, for the administrator and for hackers who want to improve or enlarge the system. Our programs and scripts are published under the GNU license, but please note that SMART, TiMBL and perhaps other programs are published under different conditions, although the source of everything that is directly related to Open Boek is available.

The end user will want to skip the technical details and only read section 2. This is why we put this section right below. The administrator should read the two following sections about installation and indexing. If you want to change the system, or want to change how it works, read everything.

Version 0 and version 1

Version 1 differs from version 0 for the most part in that the individual html-files are discarded in favour of stand-off organisation, where tokens and tags are stored in different files, and are only combined at display time. This should improve the speed of indexing. Also, the directory system is overhauled, so that a single installation of Open Boek can access several

databases. Finally, we added an annotation tool (see section 5) so that the user can create or tune the MBL data for his own databases.

DISCLAIMER No warranties are given as to the performance of Open Boek and its useability in certain areas. This manual naturally lags behind the development. Differences between the description of the system on these pages and the real thing may and will occur. Your Mileage May Vary.

more

2 User manual

Here we describe the user interface to Open Boek and the details of retrieval within the system in some detail. Apart from this interface, there is also an automatic way of putting queries to the system. This will be described in more detail in section 3 on page 23.

2.1 Selecting a database and simple retrieval

When Open Boek is opened by entering the URL in a browser, the system first lists the available databases and their state of indexing. You select one (unless it is marked as not indexed) and the browser will display the search interface (see figure 1). If the database is marked as 'indexed on pages' only, you can only list and display the files (with the special query 'filelist') but not search for keywords, place names or chronology.

Retrieval in Open Boek is very simple: just type the keywords in the space provided and press 'submit'. After a few moments you will be presented with a list of links that point to pages or documents that may be relevant to your query. Click on the link, and you will see the text of the page. Words or phrases in that text that caused the document to be flagged as relevant, are in red. It is possible to search for keywords, for timespans or for geographic locations, and for combinations of any or all three *semantic concepts*.

For every concept there is a separate inputfield (see 1). There is also a simple syntax to enter timespans and locations directly in the first inputfield: see the paragraphs 2.2, 2.2, 2.3 and 2.4).

Finally there is a reserved word: *filelist*. If you enter the word *filelist* as keyword, you will get a list with *all* documents in the database.

2.2 Keyword search

Keywords are just typed in the first inputfield, without operators such as AND or OR. Never forget that the more keywords you enter, the higher the chances to find relevant information. Overload is largely avoided by ordering the results of your query on estimated relevance. See below for an explanation of the three ways they can be combined, weighted and sorted.

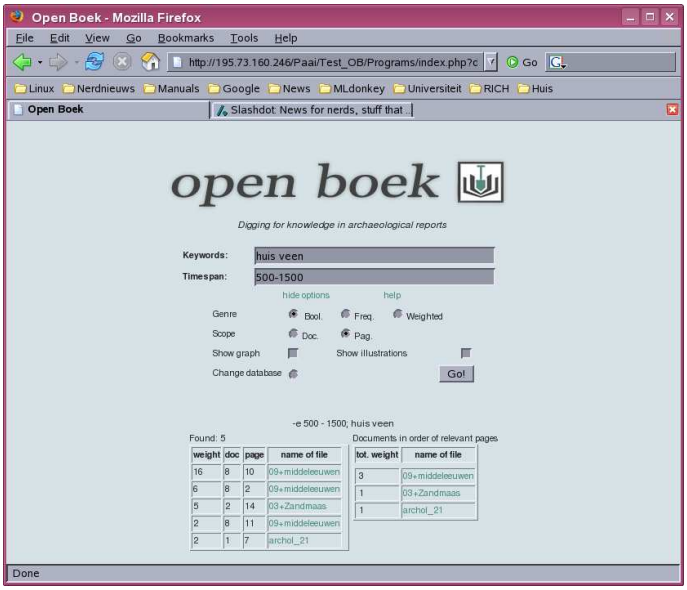

Figure 1: User interface ('show options' active)

Wildcards

A recurring problem in keyword retrieval is that of *homonyms*, words that are spelled similarly, but mean very different things, like 'bow', which may be either the front part of a ship, an instrument to shoot arrows or the act of bending before a king[1]. That is why we encourage you to describe your information need in several words: 'bow waves sea' will bring you to pages about the nautical meaning, whereas 'bow arrow' will get you to Robin Hood. That may look obvious, but research has shown that the average query on e.g. Google is shorter than two words (1.7 to be precise), and are for the most part four letters long.

Dutch, english and most other European languages use suffixes for plurals and other variations. To avoid typing in all variations, you can just type the beginning of a word, followed by an asterisk, and all variations will be included in the search. So 'bow*' as query will get you 'bow', 'bows', 'bowing', 'bowman' but also 'bowl' and 'bowel'. In the same vein, the point ('.') is used for a single character: 'd.gger' will expand to 'dagger' and 'digger'. Of course you can combine both wildcards.

[1]Please understand that OB is trained on the dutch language, but in this manual we have translated all dutch examples to their english equivalents

Relevancy

As we said under 'keyword retrieval', the list with links is sorted according to relevance, but what exactly is relevance? The answer is that we don't know. Or rather: relevance varies so wildly with the needs of the user, that it is very difficult to capture. Open Boek offers three different ways to rank the retrieved documents on estimated relevance:

Bool. We already mentioned the problem with homonyms. But apart from that, if you are interested about information about Tom, Dick and Harry, should the system assume that the pages where all three names Tom, Dick and Harry occur together will be more interesting to you than pages with only Tom and Harry? Yes, that seems obvious. Open Book will indeed assume that this is so and offer **Bool.** as the default option. But there are other options available, and you are encouraged to use them.

Freq. The option described above does not take in account the frequency of the individual words. But why would we want to do that? Because we may assume that the more often the word 'Harry' occurs on a page, the more important the concept is (for that particular page). If you select **Freq** as option, Open Boek will take the frequency of the keywords into account when it ranks the pages on relevance. Interestingly, it now is possible that pages with many Toms and Dicks, but without mention of Harry, wil rank as more relevant than a page where all three, Tom, Dick and Harry, are mentioned just once.

weighted The third option, **weighted** is very subtle, and will return unexpected, but sometimes very sophisticated results. When the 'information value' of the individual word on the individual page is computed, it not only looks at the number of times that the word occurs on the page (the frequency). As you can see for yourself, articles like 'the' and 'a' occur very often on *every* page, but they certainly have no high information value. The very fact that they occur on *every* page makes them very uninteresting.

When you search with this option, Open Boek will not only look at the frequency of the words on the page, but also at their frequency on *other* pages, dividing the frequency of the word (tf) by the number of documents it occurs in (df). This called the $tf.idf$ weight. For a more complete treatment of such weights see [7].

In the $tf.idf$ variant that we use here, the length of the individual page is also considered. Again this has interesting consequences. If the keyword happens to be on a very short page, it is marked as more important than when it is accompagnied by a herd of other words.

All these esoteric tricks and twists cause 'interesting words' to rank higher than relatively uninteresting words and as we said before, it is a good idea to experiment with these options.

1.	`cat dog horse`
2.	`1200-1400`
3.	`@1200-1400`
4.	`1200-1400; cat dog horse`
5.	`middle ages - second worldwar; cat dog horse`
6.	`1200BC-1000BC`
7.	`amsterdam(20)`
8.	`1200-1400; cat dog horse amsterdam(20)`
9.	`den_bosch`

Table 1: Valid queries in Open Boek

In picture 1 you will see the results of the search in two tables. The lefthand table shows the individual pages, and the 'weight' of that page. The righthand table shows the list of *documents* ordered on the number of pages that contain one or more 'hits'. The links will cause new windows to be opened with either the page that is referred to (lefthand table) or with the first page of the document (righthand table; see fig. 2).

If the text has been extracted from a pdf-file, you can inspect the either the original page of that pdf-file or the complete file by clicking on the button **pdf (pag)** resp. **pdf (doc)** in the upper left frame of the window where the document is displayed. Here you also will find navigation buttons to browse through the complete document. Individual pages within the current document are accessed by clicking in the lefthand frame. At the righthand side you will see a similar frame. Here you can navigate the list with 'hits', pages or documents that conform to your query. If the frame is too narrow to display the title and the page, please use the interface of your browser to adjust the width of the frame.

Strings in the text that are relevant to the query are in red. Strings that are not relevant, but that are recognized as geographical or chronological expressions are in blue.

You will observe that the HTML-file is not always well-aligned with the original 'image' of the page or that ugly overlaps or jumps in the text are visible. This cannot be helped without major surgery, and precisely for that reason we make the pdf-file also available to you. But in most cases the problems with rendering are minor or not even visible.

2.3 Chronological search

Apart from searching by keywords, it is also possible to search on chronological dates. Indeed this is one of the reasons why you would use Open Boek. Searching on dates is as simple as entering a range of years (in arabics) or the name of an era in the field provided. For the impatient, the ';'

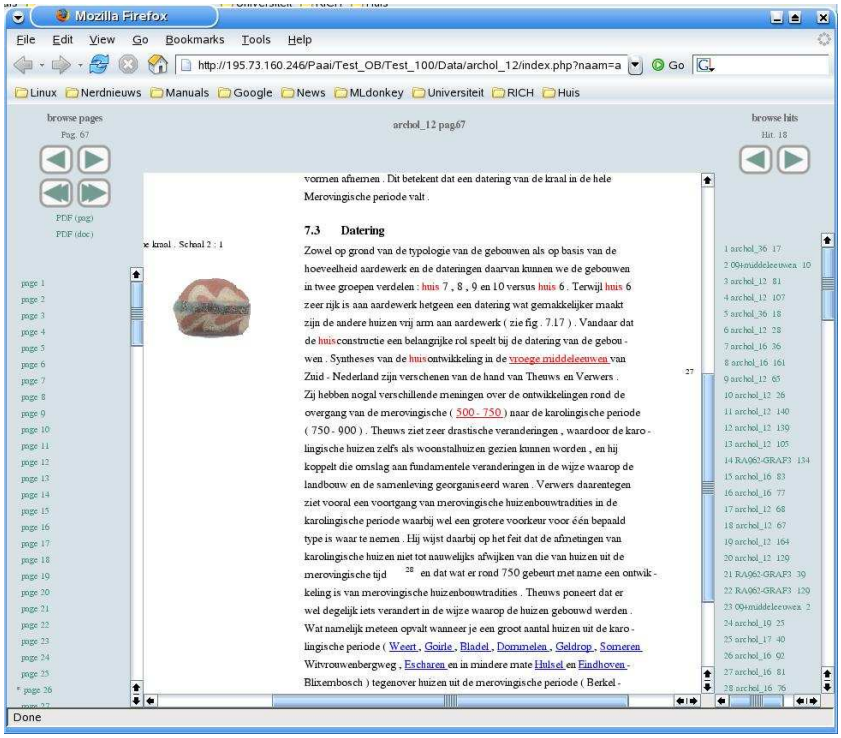

Figure 2: Display window

operator can be used to enter period and keywords in the first input field (table 2.2 lines 4, 5 and 8).

Open Boek 'knows' what time is and what years are, and will return all pages with dates that fall within the range you entered, regardless how they are written in the document. 'Twelfth century', '1100-1200' and '+XII AD' and its variations should all be recognized.

By default a range in the document should fall entirely within the period you entered, that is: if you enter 1000 to 1500, it will *not* return pages with 'middle ages'. This is because the middle ages are defined as 500 - 1500, and to retrieve them, you should enter a start date equal to or less than 500 and an end date equal to or greater than 1500. However, the *late* middle ages are defined as between 1000 and 1500 and that will be retrieved, as will be every period or individual date between 1000 and 1500 inclusive (see below for how such names of eras are recognized).

The operator @ (the 'at'-sign) changes this behaviour (table 2.2 line 3) . If a timespan is preceeded by this sign, a document will be flagged as a hit if a period in the document starts or ends in the timespan indicated by the query. '@1000-1500' will return all timespans that begin or end in that period, so now the middle ages will be retrieved.

In table 2.2 line 5 we have shown that you can enter a named period in a timespan; in line 6 the use of BC is demonstrated (BP is allowed too, where present is 1950). You can inspect a rudimentary list with named periods in *eras.rc* (table. 3). Modifications and extensions of this list should be

left to the administrator (see section 3).

The recognition of chronological dates is a function of so-called artificial intelligence, and like human intelligence it will occasionally be wrong. In most of those cases where it errs, other numbers in the text are wrongly marked as years.

2.4 Geographical search

Names of cities, villages and other geographic entities obviously can be searched as keywords. However, if you want to make use of features as distance or area search, you need extra tools.

- Distance search. The location (e.g. Amersfoort) is considered a point, and you can search for other points within a circle with a given radius. There is a separate inputfield for such searches, combined with a dropdown menu for predefined distances (5-10-15-20 km). You can also enter the location in the keyword field and add the distance between parentheses: "Amersfoort(17)" (see table 2.2 lines 7 and 8). There is also an opportunity to enter coordinates in stead of a geographical name.
- Area search (not yet implemented). The location is a polygon, and the search is for coordinates that lie within that polygon. Open questions are how the polygon is stored in the index, and how a point inside that polygon is defined.
- Disambiguation of geographical locations (not yet implemented).

Finally, Open Boek already recognizes spatial coordinates and is able to display the corresponding Google Maps. To do this, you just click on the link and Googlemaps will open in a new window. Of course, your administrator must have Googlemaps enabled on your site.

Important! Open Boek will try to ignore place names in literature references by default. This is because of the fact that the publishing information almost always contains a place name. Therefore you can not use this search feature if you expect to retrieve place names in booktitles, and you will have to enter such place names in the keyword field, which will show you *every* occurrence of the word.

Finally, spaces in place names should be replaced by underscores; use "Den_Bosch" in stead of "Den Bosch".

2.5 The ABR (discontinued)

The ABR or Archeologisch Basis Register [1] is a register with dutch archeological terms that includes a simple classification in type, material and time. In a similar way as with place names, Open Boek can recognize and index the terms from the ABR separately.

ALG.	SPEC.	GROEP	MATCAT	BEGIN_P	EIND_P	BEGINJ	EINDJ	OMSCHRIJVI
DAKPAN	—	BOUW	KER	ROMV	NTC	-12	1999	Dakpan
DAKPAN	DAKTGLG	BOUW	KER	LMEB	NTC	1250	1999	daktegel, geglazuurd
DAKPAN	DAKVERS	BOUW	KER	LMEB	LMEB	1250	1499	dakversiering (figuraal)
DAKPAN	IMBREX	BOUW	KER	ROMV	ROML	-12	449	imbrex/vorstpan
DAKPAN	TEGULA	BOUW	KER	ROMV	ROML	-12	449	tegula
DEKSEL	—	VXX	GLS	ROMV	NTC	-12	1999	Deksel
DEKSEL	—	VXX	KER	NEOVA	NTC	-5300	1999	Deksel
DEKSEL	—	VXX	MBR	ROMV	NTC	-12	1999	Deksel
DEKSEL	—	VXX	MFE	ROMV	NTC	-12	1999	Deksel
DEKSEL	—	VXX	ODB	ROMV	ROML	-12	449	Deksel
DEKSEL	—	VXX	OPH	PALEO	NTC	-350000	1999	Deksel
DEKSEL	IS66A	VXX	GLS	ROMV	ROMV	-12	69	Isings 66a
DEKSEL	IS66B	VXX	GLS	ROMV	ROMV	-12	69	Isings 66b

Table 2: Part of the ABR (slightly edited)

2.6 Other options

Apart from the different ways to weigh the keywords, there are some other options visible when you select "show options".

- A KWIC index or KeyWord In Context Index shows the keyword in its context. In Open Boek the context is rather arbitrarily set on 40 characters before and after the keyword. More important is that the searching algorithm for the KWIC index function does not depend on the index of (single) keywords, but scans the full text of the documents. Therefore it is possible to define a query that includes spaces and other interpunction (but note that all interpunction should be separated by a space. If you want to search for a single word using the KWIC index, surround it with a space on either side. At the end of the table with kwic index results, you will find a link to download the kwic index for later reference.

  The scanning of the full text may take some time on large document collections. After the first scan, the files reside in the cache, and subsequent scans during the same session go much faster.

- **Scope** Docs or Pags. The default is Pags, which means that searching will use the pages in the document as unit, and that keywords are weighted according to their co-occurence on the page. In the other case, the Documents will be the units and the results will be weigthed on the co-occurence of the keywords in the document..

- **Show graph** Activating this option wil cause a histogram to be displayed, with the frequency of the individual years in the pages found (see picture 3). Periods are expanded, so that 'middle ages' will cause all years between 500 - 1500 to be incremented by one. In this particular database interest seems to center on the years between the beginning of the iron age in Holland and the end of the middle ages. You will observe the very human tendency to gravitate towards 'round' years, such as 500 or 1000.

- **Show illustrations** If the display of illustrations in HTML-files is turned off, you can turn it on again with this option. The default is off, unless overridden by `ill_zichtbaar=Y` in the Database.rc file.
- **Change database**. Returns you to the first page of Open Boek, so you can select a different database.
- **Administration**. Starts the administration interface for the creation of new databases, indexing and similar activities.

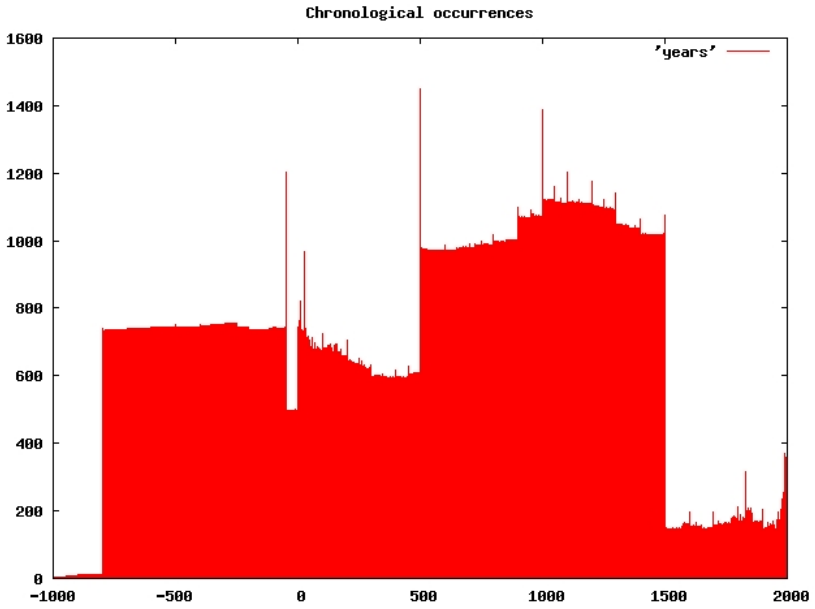

Figure 3: Histogram of chronological references between 1000 BC and 2000 AD

2.7 The index server

It is possible to submit a batch of pdf-document for indexing on our server and to retrieve the indexes for your own use. Refer to the Open Boek homepage for directions.

3 Installation

This section covers the installation and administration of Open Boek.

Open Boek runs as a collection of scripts under a http server such as Apache. For these scripts and the infrastructure you must have a Linux system available, because the Microsoft environment does not support all necessary tools. The administrator of the system should have some elementary knowledge of Unix systems, know how to install new software, use the command line interface and have the authority to change permissions. It is possible that some of the third party software has to be (re-)compiled.

We will describe in detail the steps that will be necessary to index the files in the Database-directories. There is a web-interface available with as URL http://.../admin.php. You will need a password to enter this URL: for the moment this is 'admin'. If you want to change it, you will have to do this in the source of admin.php.

Requirements

The software requirements of Open Boek are:

- a modern Unix system, such as Linux, including Apache and PHP. We used SuSE 10.1.
- the system files of Open Boek, available as a compressed tar archive[2].
- the pdf to html convertor, *pdftohtml* version 0.36[3], also in the SuSE distributions.
- a program to split a large pdf in its separate pages: pdftk[4], also in the SuSE distributions.
- a plotting program: gnuplot, also in the SuSE distributions.
- a compiled version of the venerable [6] SMART retrieval system, version 11.0[5] from 1993. A linux binary can be found in the openboek archive[3]; a clean compilation is not for the faint-hearted. In a next version of Open Boek we may distribute an alternative indexing and retrieval engine.
- a version of TiMBL 5.1.0 [2] [6].

For our development we used a HP compact with a pentium 4 2.60 Ghz and 790 Meg RAM memory, running Linux (SuSE 10.1). A typical database like the RDMZ database consists of 750 pdf-files totalling 1.7 Gb of data. The first conversion, from pdf to HTML and tag-files, takes 2.5

[2]http://www.referentiecollectie.nl/Openboek/openboek.tar.gz
[3]http://pdftohtml.sourceforge.net
[4]http://www.accesspdf.com/pdftk
[5]ftp://ftp.cs.cornell.edu/pub/smart/smart.11.0.tar.z
[6]The source is available at http://ilk.uvt.nl, but you will have to compile it yourself.

```
#data used by tagger.pl:  use two words max for eraname, use tabs.
#Era :  Begindate Enddate
Second worldwar    :  1940 1945
Late middle ages   :  1000 1500
middle ages        : 500 1000
Roman period       : -50  400
...
```

Table 3: The eras.rc file

hours. The indexing for keywords is very fast (one or two minutes); the indexing of chronology of this database takes two days (but we are working on accelerating this task). The total disk storage then is 4.2 Gig, but there is a small amount of redundancy and superfluous data that could be deleted (about 100 Megabyte) and if you use the link option, the original pdf-files will not be copied.

Important: All directories in use by Open Boek, including the Database-directories, should be read-, write- and executable for your http-server. On a Linux system the http-server will generally be user *wwwrun*. You can also assign a group, e.g.: 'users' that wwwrun and your administration user belong to, so that you can inspect and change scripts from the command line, if and when needed.

3.1 Files and Directories

If you have the system up and running and have created all indexes, you will find the following directories (we will call the directory where Open Boek was installed originally 'home') as depicted in fig. 4:

1. (home). here the Programs directory, the Coords directory and the database directories are stored. It also contains default example files for the MBL machine (time_examples.ann.dutch and loc_examples.ann.dutch), a list with places and coordinates (plaatsen_coordinaten.txt) and a default eras.rc. Also the "openboek.rc" is stored here.
2. Coords. A directory with scripts to access Googlemaps for coordinates. For every directory with such scripts a separate license must be obtained from Google, although for the moment (2007) this is without cost.
3. Programs. As we said, the directory where the programs for Open Boek are stored and under which three other directories reside: Specs, Data-php and Icons:
   - Icons. The directory where the icons and other images that the system needs, are stored. You can also find the style sheet for the interface here.
   - Specs. A directory where some special files are stored which are needed for Smart and other utilities.

- Data-php. The php files that are needed to display the documents.

4. (Database). For every database there will be an individual directory with a corresponding name. We will use the generic name 'Database' for now. Here the files, specific for that individual database are kept, and here also will be written a lot of logfiles when indexing or querying that database. In the Database directory also some files with particular options are stored. The most important is "Database.rc" where individual settings for that database are stored. The files "Database.jpg" and "Database.txt" may also exist. These files are displayed if you want a visual or written description of the database.

   After indexing the following subdirectories will exist in the Database-directory:

   - Docs. The SMART indexes for the retrieval of complete documents.
   - Pages. The SMART indexes for the individual pages.
   - Timeloc. The directory with indexes to retrieve chronological and geographical data.
   - Data. The location of the pdf-files and tag-files.
   - Data/(Documents). A series of directories, each corresponding to a single document. The name of the directory is the name of the original document, without its extension. When we refer to a directory 'Document', we mean one of those directories. Every document is split in pages (if and when possible) and every page is split in functional files: one for the tokens, one with tags for the layout, one with chronology tags and so on. Also, some php-files that combine those functional files into a coherent html file, and that govern navigation are copied from the directory Programs/Data-php and stored here.

Almost all of these files and directories will be created automatically, either when unpacking the Open Boek distribution or when creating and indexing a database of documents.

3.2 Preparation

Again note that the home directory and all directories under it should be rwx for the http server and for the administrator.

Step 1: install and prepare Open Boek

Unpack the Open Boek distribution somewhere in the document-directory of your WWW-server. We assume that SMART and TiMBL will be resident in /usr/local/bin and we have prepared defaults for that particular case. See also the variables 'timblpath' in *classify_time* and 'smartbin' in

```
# complete directory open boek
open_boek_root_dir=/Open/Paai/Test

# open boek directory minus the 'document root'
open_boek_dir=/Paai/Test

# your hostname
hostnaam=http://www.referentiecollectie.nl

# preferred language of the interface
lang=EN
```

Table 4: The openboek.rc file

smprint, *index_smart* and *query_smart*. You should also have *pdftohtml*, *pdftk* and *gnuplot* somewhere in your path.

After unpacking, first, edit the 'openboek.rc' file. This file is a small text file with some data that Open Boek should know about (see table 5). Essentially those are the name of the server and the location of the Open Boek programs and scripts. Other things, such as preferred language for the interface are also changed here, but for most variables reasonable defaults exist. Note that the hashmark (#) precedes comments, that are not interpreted by the system.

The important items in 'openboek.rc' are *open_boek_root_dir* which should point at your absolute Open Boek directory, open_boek_dir, that points to the directory relative to the wwwserver and *hostname* which should contain the hostname of your computer, preceeded by 'http://'. If you want to use a different language for the interface, add a variable *lang*. Dutch (default) is 'NL', english is 'EN'. Other languages can be added, but you should create and edit separate dialogs- and help-files in that language. If you want to add, e.g. german, you would choose 'DE' as the value of *lang* and create the 'dialogs.DE' and 'help.DE.html' files as translations of their dutch and english counterparts.

Nota Bene: the language of the interface is *not* necessarily the language of the database. If you want to add a database with documents in a language other than dutch, please refer to subsection 3.6.

At this point you should also have registered the Coords-directory with Google, if you want to use Googlemaps. If you use SMART and/or TIMBL, see to it that you have read the license agreements, and have installed the binaries in /usr/local/bin. The same is true for pdftohtml. and pdftk.

Step 2: select the documents

With an ASCII editor, create a list of the pdf-files or html-files you want to include in your database, with complete path information. It is a good idea to move this file to your Open Boek home directory and keep it there. Please choose your pdf-names so that no spaces, commas or other special

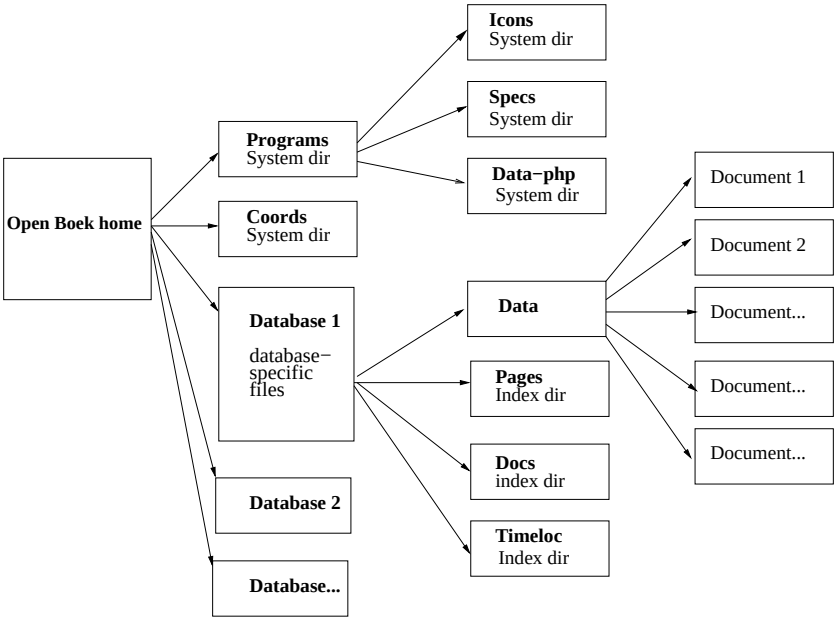

Figure 4: Directory structure

characters are part of the filename (and change the original name of the pdf-file if necessary).

Step 3: prepare the database

Open de URL http://whatever/your/open_boek_dir/admin.php (protected with password) and select the uppermost option (Create new database)(see 5).

A new screen is displayed (6): give a name for the new database that starts with a capital and the file with pdf-files. There are a few parameters that you should know about.

The first is whether you want the pdf files copied to the Open Boek structure, or just have them linked there. The default is linking; but if you want to burn your database on a CD, you will want to copy the original pdf-files.

The second is whether Open Book will try to recognize pages that contain literature references. Such references almost always contain place names and years, but such data are mostly 'uninteresting' as search argument. For instance: many archeological texts are published in Amersfoort; and such occurrences will strongly interfere with a search for archeological finds in or round Amersfoort. The default is therefore to ignore literature lists.

Then, you can protect every database with a password. This password

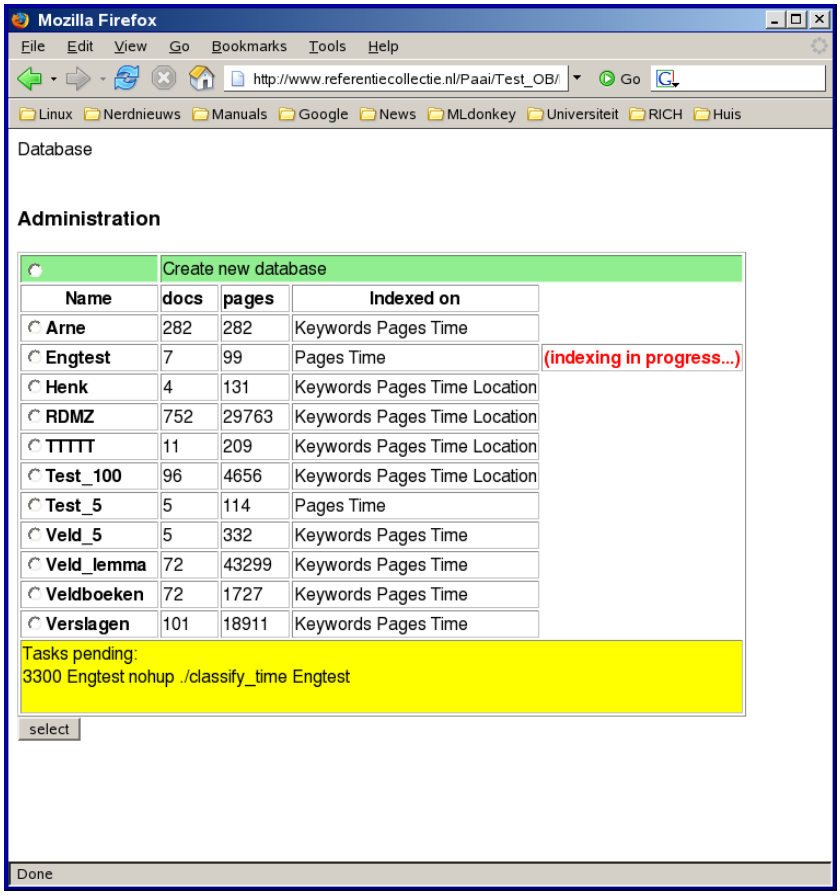

Figure 5: Menu 1 for database administration

is stored in plain text in the database.rc file, so it is not a very strong protection. (passwd)

Now press 'submit'. A new directory with the name of your database is created and the pdf files from the list will be copied (or linked) to their appropriate sub-directories under Database/Data. This can take some minutes for very long lists. It then displays the list of pdf-files at their new location.

Under the surface the following actions will also have taken place:

- A number of specification files for SMART are copied from the Specs directory to the Database-dir.
- The file 'eras.rc' is copied to the Database directory. This file contains named chronological periods (see fig. 3) and you should edit it according to your needs. Of course you can add new periods at will, as long as you conform to the examples: a colon between the name and the years, and white space between the years. Years before christ are preceeded by a minus sign. The language of eras.rc should match the language of your documents.
- If you want special features for this database, a file 'Database.rc' should exist in the Database directory. This file is an extension of the 'openboek.rc' file so that variables specific to that database can be defined, e.g: 'filecopy' if you want to copy the files in stead of linking

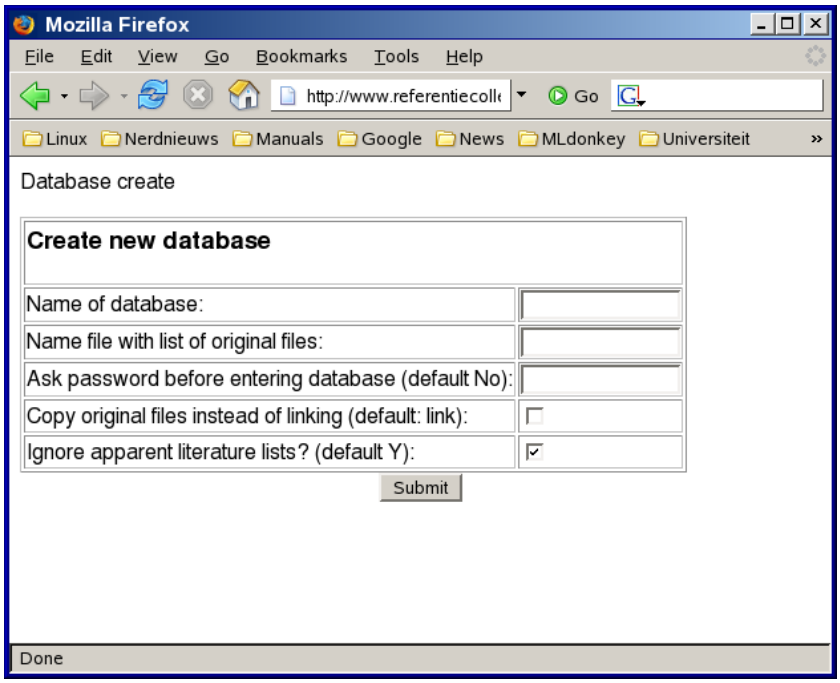

Figure 6: Creating a new database

them. This file is created automatically. here also the variable LANG is specified if the language of the documents is other than dutch.

- There will be a file created called 'Database.lst' (or whatever the name of your Database directory), that contains a list with the location of the documents, number of pages and some related information. See table 6.
- If in the home-directory files exist with the same basename as the Database-directory, they will be copied also, but you can insert them at a later date, as they are not compulsory. Such files may include: Database.jpg, for the logo (about 550 x 175) and Database.txt for a short description.

Step 4: creating the text- and tag files

Reload the page with the administrative interface. Your new database should now be visible. Select it, so that the menu in figure 7 is displayed. you will observe that the actions you can perform on every database are governed with a few buttons. Every database can have different indexes activated: they are recognizable by the fact that the text 'already done' is displayed behind the appropriate checkbox. Nevertheless, you can 'redo' such actions, although this is only useful if you want to experiment with the system.

```
# password
passwd=apekool

# ignore pages that look like bibliographies (Y or empty)
ignore_lit=Y

# local options (Y or empty)
local_options=

# display illustrations (Y or empty)
ill_zichtbaar=Y

# language of the database (NOT language of the interface)
LANG=dutch
```

Table 5: The database.rc file

If indexing is in progess, you will be notified by the fact that this is indicated in red. Also, in the yellow box at the bottom, the tasks that are currently running, are displayed. It is generally a good idea not to start new tasks when the yellow box is in evidence.

When you select a database, Open Boek will check if the conversion from pdf to html by pdftohtml has already been done; if not it will proceed to do so first and return when it has been completed. In this step, which may take some time (approx. three or four pdf-files in a minute) OB will convert the contents of the pdf-files to HTML, images (png-files) and other relevant material, notably the token-lists and the taglists. The script 'prepare_data' sees to this conversion. It then reads the HTML-files that were created by pdftohtml, and will create three separate files: one for the tokens of the text proper (doc-x_tokens), one for the interpunction (doc-x_interp) and one for the HTML tags (doc-x_taglist). The 'x' in the filename stands for the pagenumber. From now on, Open Boek will use these files to reconstruct the html-files at query time, and the original HTML-file can be discarded. In the doc-x_tokens-file every token is stored on a line of its own; in the doc-x_taglijst and later in the chronological and other tag-files, every tag is preceded by a number that refers to the linenumbers of this doc-x_tokens file.

When you return to the administrative interface, and no red text is visible, you may proceed to step 5: indexing.

Step 5: indexing, specifics

The 'Docs' and 'Pages' checkboxes need no special preparations, although you may add or edit the 'stopwoorden' file in the Database-dir. The keyword indices will be prepared by SMART. This task only takes a few minutes, (longer for large databases) after which you can use the advanced keyword search features.

For the indexing of chronological expressions, you need a file with

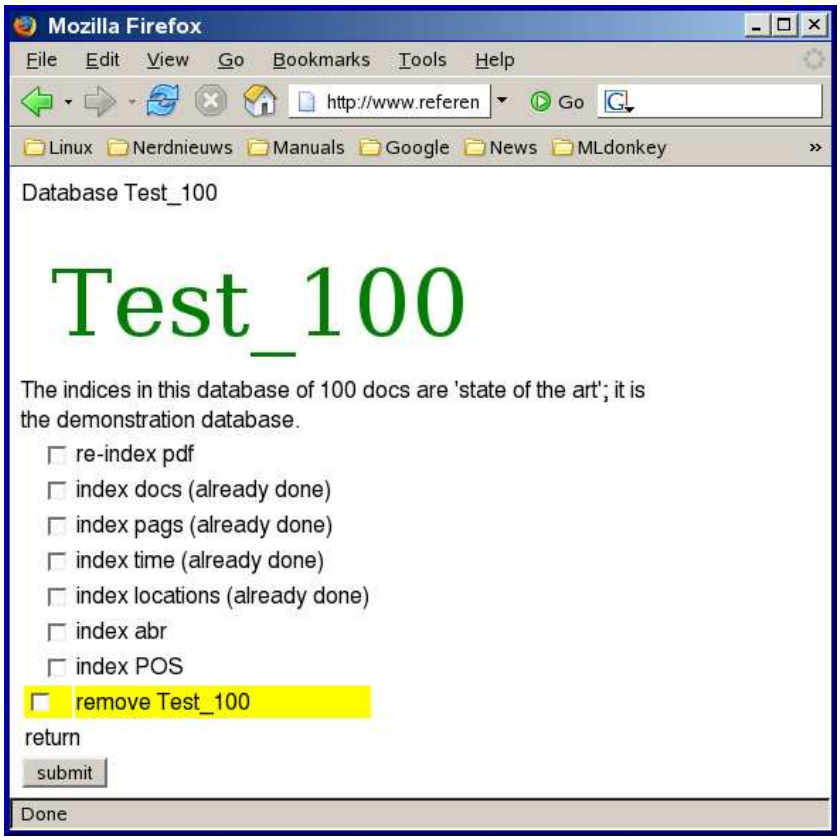

Figure 7: Menu for individual database administration

tagged examples for the language of your database. The same is true for the creation of geographical indices. These files reside in the home directory of Open Boek. The file with tagged examples for dutch is called 'time_examples.ann.dutch'.

Select the appropriate checkboxes and wait. Large databases can easily take two or three days to complete the indexing.

Nota Bene: All actions in Open Boek leave logfiles in the Database directory. See the section 4 on the names of the logfiles and when they are created. At this stage of development, the logfiles are overly verbose.

This ends the instructions on how to create and index an Open Boek database.

3.3 Some notes on document file formats

Open Boek supports both pdf and html formats. If you start with other formats, convert them to either pdf or HTML, but note that you need a textual representation of your document in the pdf-file.

pdf is logically structured as paged documents, and OB will take those pages as units vor indexing and display.

HTML has no page structure. If you want to paginate HTML-files, insert the line <!- - pagina - -> (html comment) where you want your pagebreaks. OB will put <body>...</body> tags around the individual pages, otherwise it is your responsibility to see that the HTML within the

pages always is consistent, that the tags are balanced etcetera.

By far the largest portion of the documents in our collection of about two thousand reports of approx. fifty pages each) were originally typed on paper, and later scanned, OCR-red 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 pdf-representation. Instead we convert the contents to HTML. However: this gave rise to the following problems.

1. One alternative, the omission of the image of the page, and the display of only the ASCII text as HTML gave the opportunity of highlighting and links, but omitted most visual content such as images and most formatting.
2. The second option consisted of the projection of the HTML-ized ASCII over the image. This combines highlighting, links *and* visual content, but the result in the browser often looks messy.

Another large portion of the files was already written using a wordprocessor and stored as PDF. Such files translated relatively easy in HTML, combining highlighting, links and images. Still, the rendering of the fonts is not always satisfactory. In any case you can switch from one method of display to the other.

The default in Open Boek is (1). If you want to change the default, In the database directory Database exists a file 'Database.rc'. In this file, you can put the line `ill_zichtbaar=Y`. In that case, the default will be that the illustrations are visible.

Tables and other artefacts

One of the problems with the conversion program that we used is that the resulting HTML is divided in lines (in the sense of one or more words on the same level), and that every such line is only marked by its position on the page and its font. Subscripts and superscrips are not considered part of the line; they get individual tags for font and position, after which a new 'line' is started. Every information about e.g. the line being part of a table, header or caption, is lost. A similar problem exists if the text is made up in columns; our programs do not recognize the columns but read the two lines as belonging to a single line. These problems are not solved at this moment.

Microsoft files

A third group of documents consisted of hundreds of reports written by individual archeological bureaus. These were stored on as many CDs and

almost always produced by 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 timeconsuming process of analysing the contents by hand, defeating the purpose of *automated* indexing and retrieval. But even if the 'central' document could be identified, Microsofts OLE framework often prevented extraction of the relevant data, at least with the tools that we used.

Another unexpected result of the Microsoft way of doing things was that we often found text or pictures in a Word file that were normally not visible, and certainly not meant to be visible, such as corrections, annotations and remarks, deleted pictures and so on. This can lead to embarrasing situations.

3.4 The index files

The 'Timeloc' directory contains the 'tijdlijst' index and the 'loclijst' indexes. The 'tijdlijst' depends on the existence of the machine learning components TiMBL and a database with examples. In the Open Boek distribution such a database is included ('number_examples.ann.dutch'), but you are encouraged to create your own examples. Please note that the indexing of these numeric classes is very time-consuming, depending on your hardware this can take several minutes for every document.

The 'loclijst' index tries to identify place names in the same way. It uses the file 'loc_examples.ann' for this purpose, in combination with 'plaatsen_coordinaten.txt'. By default, Open Boek will try to recognize literature references, and ignore place names in such cases. For this purpose it uses a rough heuristic, based on the ratio of interpunction, capitals and words. Please note that other pages can look like literature and be ignored. However, we found that place names on such pages generally were 'uninteresting' for the same reason that place names in literature lists are 'uninteresting'. In any case, you can always use plain keyword search to retrieve any string on such pages.

3.5 Moving databases

In the case that you want to move a complete, indexed database from one computer to another, please note the following:

If you copy the complete Openboek installation, you need to update the 'openboek.rc' file.

If you only move a database, see to it that it resides directly under the Open Boek home directory (like all other databases). Then change the ownership and group to wwwwrun:users, or whatever is valid for that host. The permissions should be u+rwx for all directories and files, or if you want to experiment, ug+rwx.

If the original pdf-files are linked to, see that the link is accessible from the new directory.

The original path information for every file is found in the files Database/doc_loc and in Database/Database.lst. You will have to load these files in a text editor such as emacs, and replace all original paths by the new path. Then, you will have to reindex the keywords (Docs and Pages) as above.

You do *not* need to reindex the Time index or the Location index.

3.6 Documents in other languages

Although you can easily change the language that is used for the interface (see subsection 3.2), it is more difficult to prepare Open Boek for documents in different languages.

The first restriction is that you can only do this for separate databases. Mixing e.g, english and dutch in one collection will not work.

After that, you should realize that the really language-dependent modules are those that select the cases for the machine learning part. Let us take the recognition and extraction of chronology data as an example. The modules that detects potential chronology-related phrases are paai_tag_time and num_pick. In these modules, we have three functions:

1. The first is to detect roman numerals and convert them to integers. This will not have to be replaced when you change from e.g., dutch to english. The source is in eval_roman.awk
2. The second function translates cardinals and ordinals to integers. Obviously this needs to be taylored to every language you want to use. Sources for english and dutch are in eval_cardinals_dutch.awk and eval_cardinals_english.awk.
3. Finally, there are some heuristics expressed as rules. These too are dependent on the particular language. They are implemented in paai_tag_time and num_pick themselves.

The system reads the database.rc file and extracts the parameter LANG=... If this does not exist, dutch is assumed.

The next ting to do is to extract about 10,000 examples of potential chronology-related phrases from a number of typical documents and categorize them by hand, possibly using the annotator described in section 5. The annotated lines are called time_examples.ann with the language as suffix: e.g. time_examples.ann.dutch. See table 10.

There exists a dirty trick to extract such lines from the databases. What you do is take an *empty* example file and proceed to create a chronlogy index. After completion, there exists a Database/Temp directory, with for every page in your database a file ending on ...txt.num. Now collect from those files as many lines as you need, and categorize them according to your system...

3.7 Automated Retrieval

The Open Boek system can be queried without using the user interface described in section 2. In that case, the query must be sent as a GET parameter to the following URL: http://.../json.php. The result will be returned in josn-format.

If you want to hack this...

4 Detailed description of the programs

This section contains detailed information on the Open Boek internals: scripts, logfiles and other stuff that you need when you want to develop your own Open Boek modules.

4.1 Prepare_data

The first program that will be run for a new database is *prepare_data*.

- (if called with option pdf) it calls pdftohtml to extract from the pdf-file the individual pages as numbered HTML-files and separate images. It also creates an index-file, called name_ind.html, and OB uses this index file to keep track of the pages.
- it extracts the text proper from the HTML-files, so that SMART can later index those files, adding the markers <PAGE...>, <DOC ...>, <TEXT> and <STOP> for the SMART preprocessor.
- it extracts the tokens from the individual pages (HTML-files) and stores them in *_token files. Dito for the HTML-tags, which are stored in the *_taglijst files and punctuation information (*_interpunction).
- it adds a number of files in every directory: index.php, knoppen.php, hitknoppen.php lijst.php, hitlijst.php and pasop.html. OB needs those files to display the HTML-files later and to allow you to navigate through the pages of the documents.
- finally it writes the files 'doc_loc' with the filenames (needed by SMART) and 'Database.lst' (see table 6) with a concordance of pagenumbers and documents to the home directory.

Prepare_data keeps a log of its actions in the database directory as *prepare_data.log*.

```
cum# pag# full pathname
 27,27,"/Open/Test/Demo/Data/Aalburg",
 98,71,"/Open/Test/Demo/Data/Aalsmeer",
125,27,"/Open/Test/Demo/Data/Aalten",
143,18,"/Open/Test/Demo/Data/Aardenburg",
...
```

Table 6: The Database.lst file, showing from left to right the cumulative number of pages, the number of pages and the complete name of the document.

filename	start	end
Aalburg-26	+19870101	+19871231
Aalburg-27	+19300101	+19301231
Aalburg-27	+19000101	+19001231
Aalburg-27	+19000101	+19001231
Aalburg-27	+19360101	+19361231
Aalsmeer-1	+19920801	+19920831
Aalsmeer-2	+19300101	+19301231
...		

Table 7: The 'Timeloc/tijdlijst' index

4.2 Creating the keyword indexes

At this point the keyword indexes can be created, after which Open Boek can already be used as an advanced VSM-based retrieval system. In the home directory, you will see a number of files, beginning with 'spec.'. These files govern the behaviour of SMART. It should not be necessary to change anything in those files, but note that if you want to use a list of stopwords, it should be called 'stopwoorden'. This file has to be present, but it can be empty.

We will assume that the binary 'smart' is copied to '/usr/local/bin'.

index_smart. This script is executed twice; once for the indexes on document level, and once for the indexes on page-level. Actually, this is redundant, so we will change that some day. The script calls smart to create the frequency- and atc (tf.idf) indexes. The results are stored in the directories 'Docs' and 'Pages' respectively. Then the script *smprint* is called to create human- readable indices (word_weights.atc and word_weights.nnn). Finally it creates the 'inverted_file' files in Docs and Pages that are used for the traditional (**Conj** way of searching.

Logs are kept in the database directory as *index_smart.log* and *index_time.log*.

4.3 The time indexes

classify_time. This script handles the recognition and indexing of chronology and other numeric data. It calls *wintok* and *numpick* to make lists of numbers in context. The script *paai_tag_time* recognizes whether the expressions are chronological or spatial coordinates and creates the *_taglijst_chron files with the timespan tags for every page. Then, *index_time* extracts the 'Timespan' information from those files and stores it in 'tijdlijst' as an index (see table 7).

The logfiles are : *classify_time.log*, *numpick.log*, *paai_tag_time.log* and *wintok.log*.

4.4 The location indexes

classify_loc. This script handles the recognition and indexing of place names from the list 'plaatsen_coordinaten.txt'. It calls *wintok* and *loc_pick* to make lists of place names in context. The script *paai_tag_loc* recognizes whether the expressions are proper place names and creates the *_taglijst_loc files with the timespan tags for every page. Then, *index_loc* extracts the information from those files and stores it in 'loclijst' as an index.

The logfiles are : *classify_loc.log*, *loc_pick.log*, *paai_tag_loc.log* and *wintok.log*. There is also a *lit.log* that records which pages were not indexed because they were flagged as 'literature'.

4.5 Retrieval

Retrieval is based on the indexes in the Pags en Docs directories, on the indexes in the Timeloc-directory and on the file 'Database.lst'. The results are written to temporary files in the Database-directory, prefixed with 'tmp...'. Every query has an unique number, so that the tmp-files can be inspected in case something unexpected happens, but all tmp_files older than 24 hrs are deleted whenever index.php is called.

- a php-script (index.php) is called in a browser. Keywords, chronological queries and geographical queries (class queries) are entered in separate fields. The intermediate results are stored in tmp-files, which then are joined.
- the script *query_smart* calls smart with a query; generally as a back-end of the php-interface script. It also can read the inverted files and perform a boolean query. 'Database.lst' is used to find the name of the document from the page. The SMART engine is used by creating a file with the commands that would be given from the interactive interface of SMART, and collecting the output from SMART in a file. Long live the Unix pipe! The logfile for this action is *query_smart.log*; the resultfile something like 'tmp_result_12345_key'.

file-page	starttime	endtime
4+Grensmaas-3	+19980101	+19980101
04+Grensmaas-5	+20040101	+20040101
04+Grensmaas-5	+19990101	+19990101
04+Grensmaas-5	-501230	+5000101
04+Grensmaas-5	+5000101	+5000101
05+natte+archeologie-10	+20020101	+20020101
05+natte+archeologie-11	+20010101	+20010101
05+natte+archeologie-11	+20040101	+20040101
05+natte+archeologie-12	+19400101	+19450101

Table 8: The time index.

- *query_time* queries the 'tijdlijst' file. It also does a last check on consistency. logfile: *query_time.log*. The resultfiles something like 'tmp_12345_chron'and 'tmp_12345_chron_tmptijd. This last file is created to create a graph with 'tijdsgraaf'.
- *query_loc* queries the 'loclijst' file. logfile: *query_loc.log*. The resultfile something like 'tmp_12345_loc'.
- The final results are written to temporary files (see table 9). It contains from left to right the weight, the absolute pagenumber, the page in thedocument and the document path. If both *query_time* and *query_smart* were called, the result is the join of both results.The resultfile something like 'tmp_result_12345'.
- The php-interface reads this file and displays the list of pages and documents.
- Each page links to the file index.php in the subdirectory of that document. This script displays the corresponding page, using the script *highlight* to highlight selected markups and where possible, to improve rendering. It leaves the following logfiles in the document-directory: tmp.html, combine.log, highlight.log, index_time.log and wintok.log.

The queries are solved as follows: the temporal, geographical and keyword indexes are scanned for matches; the matches are stored in 'tmp_result_12345_key', 'tmp_12345_loc' and 'tmp_12345_chron'. These files then are combined according to the genre of the query (boolean, frequency or advanced) and stored in the ultimate resultfile 'tmp_result_12345'.

```
tmp_chron_1200991778
tmp_chron_1200991778_tmptijd
tmp_result0_1200991778
tmp_result0_1200991778_docs
tmp_result0_1200991778_key
tmp_result1_1200991778
tmp_result2_1200991778
```

weight	doc	page	filename
0.16	20	10	/Open/RDMZ/Data/Amersfoort
0.15	20	13	/Open/RDMZ/Data/Amersfoort
0.14	42	11	/Open/RDMZ/Data/Barneveld
0.13	20	15	/Open/RDMZ/Data/Amersfoort
0.13	168	12	/Open/RDMZ/Data/Eemnes
0.12	341	63	/Open/RDMZ/Data/Leusden

Table 9: The result of a query as stored in a tmp-file.

5 The annotator

As Open Boek for its special functions depends on the existence of annotated examples, we have also added a simple web based annotation tool. It is called directly from your browser or from the Open Boek administrator interface.

To use the annotator, you must prepare a file with text windows (sequences of a certain number of words) with a focus of the feature that you want to classify and a label field for the assigned class. See 3.6 for an easy trick to do create such files from existing pdf-documents.

As an example, consider the file "time_examples.ann" (table 10)

The file has nine features. The feature to be classified is in the column 'focus' and is in our case a numeric, a cardinal or an ordinal. The purpose of annotation is to enter the correct label in the last column.

You can start the annotator by loading the URL http://.../annotator.php. Our annotator expects the file to be annotated to have the suffix ".ann", and to have spaces as separators between the attributes. This file should be stored below the Programs directory and have the name 'Annotate'. When you start working with the annotator, new files also get a number in the filename, that is incremented after every save. This ensures that you have a complete history of your efforts, in case something bad happens.

The first time you select a file for annotation, you must enter the number of classes that you will be using, and press the button 'reload'. Then, indicate the number of features in the file, the focus field and the field with the class, but the annotator will already have computed them. After the first run, the annotator will save the values you have selected in a file with a .rc suffix and reload them automatically.

The annotation is straightforward: every line presents the classes you may to assign; just click on the corresponding radio button (see figure 8). When you are tired, press one of the buttons with "Save" that occur every ten lines; your work will be saved with the next highest number.

The structure of the annotation files is written in a rc-file that has the corresponding name.

/subsectionAdding evaluation information

It is easy to apply this annotator as an evaluation tool. Given a

				focus					Label
telefoon	:	020	-	463	4848	Zeedijk	54	telefax	[Other]
AAI	's	:	tussenbalans	1	januari	2000	,	Maastricht	[E52_Timespan]
o	25	-	30m	10	-	15m	3	03	[Other]
veen	0	0	0	0	1	411	20	veen	[Other]
de	hand	.	Figuur	17	(	links	)	coupe	[Reference]
Drie	fibulae	uit	de	eerste	helft	van	de	eerste	[E52_Timespan]

Table 10: The contents of time_examples.ann

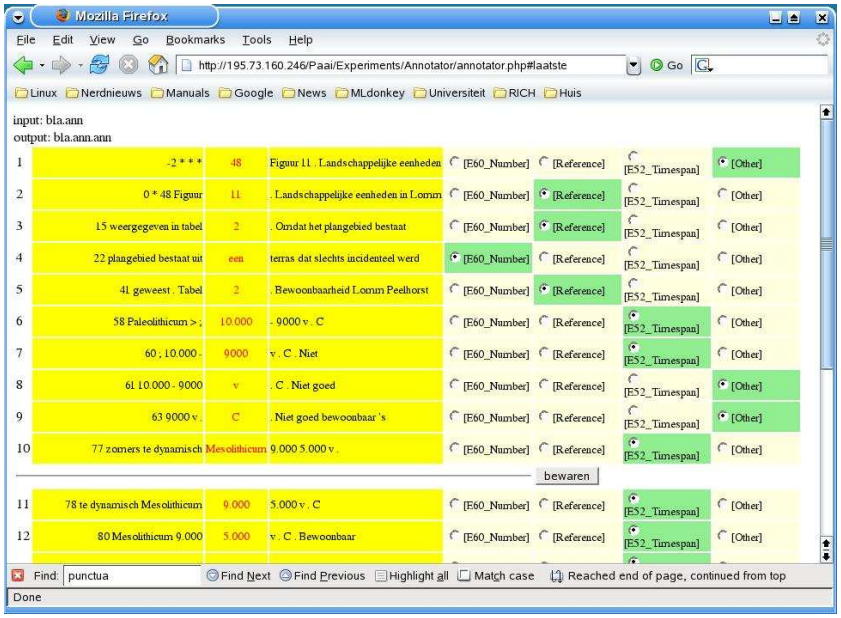

Figure 8: Annotator for time_examples.ann.

database filled with what Open Boek assumes are the correct instances for every case, you only have to add a new label field with classes like '[correct]' and '[false]' and proceed to use these labels as the new classification. You can obtain such files by collecting from the directory database/Temp all files ending on '.classified' (see also 3.6). After tagging the instances by these labels, it is relatively easy to compute the performance of Open Boek for the given documents.

Nota Bene: the annotation task often is much easier if you sort the records on the focus column or any other criterium that ranks them in sensible groups. .

6 Acknowledgements

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

No Microsoft software was used in research or production of this document.

References

[1] R.W. Brandt, E. Drenth, M. Montforts, R.H.P. Proos, I.M. Roorda, and R. Wiemer. *Archeologisch basisregister, versie 1.0*. Archis expertise centrum, 1992.

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

[3] J. J. Paijmans. Indexing texts with smart. *Linux Journal*, (36):24–26, april 1997.

[4] J.J. Paijmans and S. Wubben. 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, 2007.

[5] J.J. Paijmans and S. Wubben. Preparing archeological reports for intelligent retrieval. In Posluschny, K. Lambers, and I. Herzog, editors, *Proceedings of CAA-2007 (in press). Berlijn, Germany*, volume 10 of *Kolloquien zur Vor- und Frhgeschichte*. Dr. Rudolf Habelt GmbH, Bonn, 2007.

[6] G. Salton, editor. *The SMART retrieval system; experiments in automatic document processing*. Prentice-Hall, Englewood Cliffs, N. J. , 556 pp., 1971.

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

Appendix: the Timbl license

The TiMBL License

```
Licensing Terms

Downloading and using the TiMBL software implies that you accept the
following license terms:

Tilburg University and University of Antwerp (henceforth
‘‘Licensers’’) grant you, the registered user (henceforth ‘‘User’’)
the non-exclusive license to download a single copy of the TiMBL
program code and related documentation (henceforth jointly referred to
as ‘‘Software’’) and to use the copy of the code and documentation
solely in accordance with the following terms and conditions:

The license is only valid when you register as a user. If you
have obtained a copy without registration, you must immediately
register by sending an e-mail to Timbl@kub.nl.

User may only use the Software for educational or non-commercial
research purposes.

Users may make and use copies of the Software internally for
their own use.

Without executing an applicable commercial license with
Licensers, no part of the code may be sold, offered for sale, or made
accessible on a computer network external to your own or your
organization’s in any format; nor may commercial services utilizing
the code be sold or offered for sale. No other licenses are granted or
implied.

Licensers have no obligation to support the Software it is
providing under this license.  To the extent permitted under the
applicable law, Licensers are licensing the Software "AS IS", with no
express or implied warranties of any kind, including, but not limited
to, any implied warranties of merchantability or fitness for any
particular purpose or warranties against infringement of any
proprietary rights of a third party and will not be liable to you for
any consequential, incidental, or special damages or for any claim by
any third party.

Under this license, the copyright for the Software remains the
joint property of the ILK Research Group at Tilburg University, and
```

the CNTS Research Group at the University of Antwerp. Except as specifically authorized by the above licensing agreement, User may not use, copy or transfer this code, in any form, in whole or in part.

Licensers may at any time assign or transfer all or part of their interests in any rights to the Software, and to this license, to an affiliated or unaffiliated company or person.

Licensers shall have the right to terminate this license at any time by written notice. User shall be liable for any infringement or damages resulting from User's failure to abide by the terms of this License.

In publication of research that makes use of the Software, a citation should be given of: ''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, Available from http://ilk.uvt.nl/downloads/pub/papers/ilk0402.pdf

For information about commercial licenses for the Software, contact Timbl@kub.nl, or send your request in writing to:

Prof.dr. Walter Daelemans
CNTS / Center for Computational Language and Speech Processing
Department of Linguistics
University of Antwerp
Universiteitsplein 1
B-2610 Wilrijk (Antwerp)
Belgium

Files in the OB distribution (version 1.0)

```
Specs          :  a directory with specification files for smart,
a file with chronological eras (eras.rc), a stoplist
for the dutch language and an example bibref file.
Coords         :  a directory with scripts for Googleearth.

admin.php      :  the administrative interface
check_tasklist :  a script to see if OB-related tasks are running
classify_time  :  the envelope script for chronology-indexing
classify_loc   :  the envelope script for location-indexing
combine        :  a script to combine stand-off files into a single
HTML file
eras.rc        :  a list with time periods and years.
functions.php  :  a collection with php functions
highlight      :  a script to highlight hits in the HTML version
index.php      :  the query interface
index_smart    :  a script to index using smart.
index_time     :  a script to extract time information and index
it.
loc_pick       :  selects expressions containing placenames.
numpick        :  selects expressions with numbers or digits.
openboek.rc    :  global parameters for Open Boek
paai_tag_time  :  a rule-based postparser for chronological expressions
paai_tag_loc   :  a rule-based postparser for chronological expressions
prepare_data   :  the script that extracts the html from pdf and
creates
the stand-off files.
query_smart    :  pipes a query into smart and collects the output
query_time     :  solves chronological queries
query_loc      :  solves geographical queries
tijdsgraaf     :  draws a graph of chronological references
wintok         :  writes text in columns.
check_lit      :  checks a page if it is a literature list

dialogs.EN     : screen texts in two languages
dialogs.NL
help.EN.html   :  help in two languages
help.NL.html
time_examples.ann   :  the database for MBL based chronological
tagging.
loc_examples.ann    :  the database for MBL based place name recognition.
plaatsen_coordinaten.txt :  the database with locations and coordinates.
```

Files used for the MBL examples

01+inleidings.html
02+doelstellingen+en+organisaties.html
03+Zandmaass.html
04+Grensmaass.html
05+natte+archeologies.html
06+steentijds.html
07+metaaltijds.html
08+romeinse+tijds.html
09+middeleeuwens.html
10+conclusiess.html
11+samenvattings.html
12+Zusammenfassungs.html
13+publicatiess.html
HOP1_Gasleidings.html
AAIrap14-1.html
AAIrap14-10.html
AAIrap14-11.hml
AAIrap14-12.html
AAIrap14-13.html
AAIrap14-14.html
AAIrap14-15.html
AAIrap14-4.html
AAIrap14-5.html
AAIrap14-6.html
AAIrap14-7.html
AAIrap14-8.html
AAIrap14-9.html
AAIrap20s.html
AAOrap02s.html
AAOrap14s.html
AAOrap15s.html
AAOrap22s.html
AAOrap29s.html
AAOrap33s.html
AAOrap36s.html
AAOrap38s.html
Hanzelijn7s.html
NO1328-LOEBs.html
NO1342-BOHAs.html
NO1353-VREIs.html
RA1156-NEBENs.html
RA969-NLDAs.html
RAM_79_01_Hoge_Vaart-A27s.html
RAM_79_03_Hoge_Vaart-A27s.html
RAM_79_04_Hoge_Vaart-A27s.html
RAM_79_05_Hoge_Vaart-A27s.html
Rapport+86s.html
archol_06s.html
archol_08s.html
archol_15s.html
archol_18s.html
archol_21s.html
archol_22s.html
archol_26s.html
archol_27s.html
archol_30s.html
archol_37s.html
archol_41s.html
fratsen_1s.html
fratsen_3s.html
fratsen_5s.html
inhoudsopgaves.html
ockenburgh-jaarverslag-1993s.html
page5.html
F1+format+selectieadvies+waarderend+onderzoek+Maaswerkens.html
F2+format+programma+van+eisen+waarderend+onderzoek+Maaswerkens.html
F4+format+standaard+bepalingen+veldwerk+Maaswerkens.html
F5+format+standaard+bepalingen+uitwerken+Maaswerkens.html
F8+format+standaard+bepalingen+eindrapport+Maaswerkens.html
14+medewerkers+Projectteam+Archeologie+Maaswerkens.html
RAP+515_4100420_Eelde+Kosterijwegs.html
RAP+521_4100020_Beesel+Hoeve+Oud+Waterloos.html
RAP+558_4094100_Ede+Tuinderslaans.html
Selectieadvies+definitief+onderzoek+Lomms.html
HIO01_project_metainformaties.html

Index

Contents