SICS Technical Report T99:02
SICS-T--99/02-SE

ISSN: 1100-3154

An Experimental Digital Library Platform
A Demonstrator Prototype for the DigLib Project at SICS

by

Anette Hulth and Anna Jonsson

Swedish Institute of Computer Science
Box 1263, S-16429 Kista, SWEDEN

An Experimental Digital Library Platform

– A Demonstrator Prototype for the DigLib Project at SICS

Authors:

Anette Hulth & Anna Jonsson

E-mail:

hulth@dsv.su.se & anjo@sics.se

Published:

January 27, 1999

Abstract:

Within the framework of the Digital Library project at SICS, this thesis describes the implementation of a demonstrator prototype of a digital library (DigLib); an experimental platform integrating several functions in one common interface. It includes descriptions of the structure and formats of the digital library collection, the tailoring of the search engine Dienst, the construction of a keyword extraction tool, and the design and development of the interface. The platform was realised through sicsDAIS, an agent interaction and presentation system, and is to be used for testing and evaluating various tools for information seeking. The platform supports various user interaction strategies by providing: search in bibliographic records (Dienst); an index of keywords (the Keyword Extraction Function (KEF)); and browsing through the hierarchical structure of the collection. KEF was developed for this thesis work, and extracts and presents keywords from Swedish documents. Although based on a comparatively simple algorithm, KEF contributes by supplying a long-felt want in the area of Information Retrieval. Evaluations of the tasks and the interface still remain to be done, but the digital library is very much up and running. By implementing the platform through sicsDAIS, DigLib can deploy additional tools and search engines without interfering with already running modules. If wanted, agents providing other services than SICS can supply, can be plugged in.

Keywords:

Information seeking strategies; digital library platform; textual visualisation.

TR: T99:02 **ISSN:** 1100-3154

1 Introduction

Ever since humankind developed written language, gathering and storing the writings for future use have been a matter of course in many communities. The first libraries, as such, arose in the ancient civilisation of Mesopotamia for the preservation of administrative records (Sampson, 1985). The libraries in our age have a similar albeit extended role: to acquire and preserve books, journals, documents, etc., and to enhance the availability of these for the public. These tasks are, however, not trivial to accomplish: many libraries are rapidly growing out of storage space as the amount of publications keeps increasing; some editions may not be available at the local library; and old documents may be all but impossible to obtain. New technology has for the last decades provided libraries with possible solutions for effective storing as well as for world wide availability: digital documents require much less space than physical documents do, and can easily be distributed through networks.

The concept of *digital libraries* was formed by the research community not so much as an attempt to replace traditional libraries, but rather as a complement to them. In recent years the interest in the development and the use of digital libraries have grown radically (Griffin, 1997). This is mainly due to the advances in computing and communications technology, allowing large-scale creation, gathering and manipulation of digital information. A digital library consists of a database of diverse information and of tools for accessing, manipulating and exploring this information. The database can be a single collection or consist of multiple geographically dispersed collections. The collections are basically text based, but as manipulation of other types of media, such as pictures, videos and sound is improving, the incidence of these will increase (Smeaton, Morony, Quinn, & Scaife, 1998).

In a traditional library, a person assumes that all relevant tasks can be carried out without leaving the premises. A user visiting a digital library must also be given this possibility: to perform different tasks in one place (Hulth & Jonsson, 1998). According to a user study performed by Belkin and Carballo (1998), four prototypical information seeking interactions exist: finding a known information object; recognising useful information objects by scanning through an information resource; evaluating the usefulness of information objects; and determining the content/structure of a set or collection of information objects. By assembling various information seeking tools – each carrying out a specific task – in a single interface, the different information seeking interaction strategies can be met.

The Information and Language Engineering group at SICS (Swedish Institute of Computer Science)[1] recently started a digital library project, called DigLib. The project aims to develop an example library with real material and tools for information organisation, management and access. The tools will be developed for multilingual and multi-medial (i.e., text, sound, pictures) information access for different types of user demands. The library will, amongst other things, serve as a demonstrator prototype and be used for the testing and evaluation of different information seeking tools.

[1] SICS is a non-profit research foundation funded by the Swedish Board for Technical and Industrial development (NUTEK) and by a group of corporations: CelsiusTech Systems; Ericsson; Försvarets Materielverk (FMV); Hewlett-Packard, Sweden; IBM Svenska; Statens Järnvägar (SJ); Sun Microsystems; and Telia AB. For more information on SICS, see http://www.sics.se.

1.1 Assignment

As a part of the DigLib project at SICS, we were assigned to build an experimental platform with a front-end for search engines and other tools suitable for exploring a digital collection. The assignment comprised the construction and implementation of a tool for keyword extraction from Swedish documents. The platform was to be accessible on the World Wide Web.

1.2 Limitations

The work described in this thesis does not take into account any evaluations; neither concerning the integration of the different tools in the DigLib platform, nor of the Keyword Extraction Function. We will, however, give suggestions about how an evaluation of the latter could be carried out (see section 3.3.5). In order to perform an evaluation regarding the integration of the tools in the platform, a more solid evaluation of whether the information seeking strategies, (proposed by Belkin and Carballo, 1998) are the prevailing, must be done.

1.3 Method

The assignment was divided into four parts. These involved

1. defining the structure of, and choosing formats for, a collection of digital documents
2. installing and tailoring one existing search engine
3. constructing and implementing a keyword extraction tool for Swedish texts
4. collecting the tools in one common interface working towards the document database as well as towards the users.

The work, outlined above, was not performed in any strict chronological order, although the realisation of some tasks was dependent on others being started or even completed. We will explore each of these four tasks in more detail below.

Defining the Structure of, and Choosing Suitable Formats for, a Collection of Digital Documents

There is, to our knowledge, no universally prevailing standard for archiving documents for a digital library. When defining the structure of the digital collection, our desire was to make it reasonably easy to grasp for a human administrator. We chose to place the files in a hierarchical manner, grouped in regards to their origin. Specific requirements regarding directory structure, in order for deployed tools to work, can be met by symbolic links. The use of links gives the opportunity to bypass the actual directory organisation and can create the demanded structures. The objective, when choosing storage formats for our documents, was that they would be valid in the future as well. Another aspect was to choose formats that a majority of the users are able to view, regardless of computer operating system and software.

Installing and Tailoring the Search Engine

The deployed search engine is called *Dienst*, and it performs its search in bibliographic records, where, for example, title and author are specified (Davis & Lagoze, 1994). It was chosen to provide what Belkin and Carballo (1998) call ”finding a known information object”.

The deployment required both that the configuration files were tailored and that adjustments of the default interfaces were done. Although the system came with instructions on how to go about solving necessary (and other desired) changes, this work mounted up to quite a handful.

Constructing and Implementing a Keyword Extraction Tool for Swedish Texts

We constructed a tool for extracting keywords from Swedish documents, called *Keyword Extraction Function* (KEF). It enables users to determine the contents of the collection by viewing an index of keywords. This can, in parallel, be seen as an alphabetic index in a book, and corresponds to what Belkin and Carballo (1998) call ”determining the content/structure of a set or collection of information objects”. The KEF program excerpts nouns from documents, which are presented to the user in a list resembling a clickable table of contents. The list can be used either as a simple look-up help for writing queries, or, by selecting a word, the document or documents containing the term are listed.

By analysing a set of Swedish documents from the collection, the words considered to reflect the content were identified. The conclusion was, in accordance to Källgren (1984), that nouns are the most suitable words. To localise the nouns a tagger, assigning every word its word-class, called Swecg is used (Birn, 1995). The required modules for KEF, apart from the tagger, were developed by one of the authors using the object oriented programming language Java (Flanagan, 1997; Sun, 1998).

Collecting the Tools in one Common Interface

The search engine (Dienst), the Keyword Extraction Function (KEF) and other facilities, such as the browsing function and information about new documents, were assembled in one Graphical User Interface (GUI). The browsing function corresponds to what Belkin and Carballo (1998) call ”recognising useful information objects by scanning through an information resource”. Being an experimental platform[2], where information seeking tools will be tested and evaluated, it is of great importance that the tools easily can be added and removed. One possible alternative to solve this issue was to construct ordinary static Web pages or Web pages including for example Shockwave (Shockwave, 1999). That would, however, be an unsatisfying solution, since every change in the digital library would have to be met by additional Web pages and CGI-scripts.

A neater approach was to collect the tools in a common interface using the system *sicsDAIS* (Dynamic Agent Interaction System) (Espinoza, 1998). sicsDAIS is built so that components easily can be added, manipulated or removed without interfering with modules already running. A fair amount of effort learning about the system was however needed before it could be taken into use. The collection was, nevertheless, made available on the World Wide Web (WWW) using ordinary Web pages, while the real platform was being constructed. The Web pages included the browsing option, Dienst (with minor changes to the user interface) and information about the DigLib project.

During all stages of the work, appropriate literature studies were undertaken.

1.4 Who Contributed with what

Anette was mainly responsible for the work with Dienst and the design of the interface for the platform, Anna for the coding of KEF and the work with sicsDAIS. The rest of the work was carried out in healthy co-operation.

[2] *Experimental platform* and *demonstrator prototype* both refer to the implementation of DigLib in accordance to the assignment. From now on we will use the term *experimental platform*, or merely *platform*.

1.5 Thesis Outline

The outline of this report is as follows: Chapter 2 gives a general background to digital libraries and to information retrieval. In chapter 3 our digital library is presented: first a description of the DigLib collection, then the DigLib tools (Dienst, the Keyword Extraction Function, etc.). In chapter 4 the underlying system, sicsDAIS, is described as well as the design of the interface. In chapter 5 we suggest future work. Chapter 6 concludes the thesis containing a discussion about the performed work.

2 Background

The first section of this chapter gives a general introduction to digital libraries, how they differentiate from ordinary libraries and what research domains are involved. This is followed by a section on an extensive digital library project, namely the *US Digital Libraries Initiative*. We will thereafter look more closely into the SICS DigLib project. A short description of optical character recognition is also given. The chapter is concluded with an introduction to information retrieval.

2.1 Digital Libraries

Conceptually, traditional libraries and digital libraries are built much in the same way: the complexity of their collections are alike (journals, articles, photos, maps, etc.), although the contents in a digital library reside on electronic media; the methods of information organising, cataloguing and indexing are similar (i.e. some parts are automated and some parts are handled by humans). The technology of digital libraries attempts to transform various institutions, including traditional libraries, universities, laboratories and business, both regarding functions as well as service. The extent and pace to which this can be accomplished rely on factors such as legal issues regarding intellectual property, communications infrastructure, and user demands (Griffin, 1997).

Digital library research and development is not merely a matter of technology; social, legal and economic issues are of high importance, accompanied – through the entire development process – by user and usage studies.

Research in digital libraries can, according to Griffin (1997), be categorised in three major parts:

- systems-centred research (which concerns architecture design, scalability, interoperability, reliability and adaptivity)
- collection-centred research (focuses on building, maintaining, preserving and providing access to the digital collection)
- human-centred research (user and community needs, practices and expectations and the technology for these issues)

The label *digital libraries* has come to be used in somewhat different ways; some refer to a collection of digital information, others to the technologies that manages such a collection. Lynch and Garcia-Molina (1995) conclude that the most appropriate use of the label seems to integrate both definitions. Hence, a digital library is large-scale, organised collections of multimedia information and knowledge, and tools and methods to enable search, manipulation and presentation of this information and knowledge. This is also the definition that we have adopted in this thesis.

2.2 The US Digital Libraries Initiative

The amount of digital library projects, throughout the world wide research community, is substantial. It would be an impossible task to make fair descriptions of even a handful of them. To at least give some idea of what a digital library project can look like, the following section gives a description of the US Digital Libraries Initiative (DLI). The reason why this project was chosen, is that the expected outcome of the project probably will have a large impact on other similar projects as well as on the research area as a whole. Additionally, there are plans for a collaboration between SICS DigLib project and DLI.

The US Digital Libraries Initiative is a large-scale project run by several American universities, and is perhaps the most influential digital library project now running - at least the most visible. The first phase (DLI, 1998a) was founded in 1994 through a joint initiative of the National Science Foundation (NSF, 1998), the Department of Defense Advanced Research Projects Agency (DARPA, 1998) and the National Aeronautics and Space Administration (NASA, 1998) and is more or less completed. The goal of DLI Phase One was to improve the means to collect, store and organise information in digital forms, and make it available for processing, searching and retrieval via communication networks (Griffin, 1997). DLI does, however, not take into account the practical issues of actually building a digital library (Schatz & Chen, 1996). The project consists of six different digital library research groups, each aiming to solve different issues.

The DLI research groups are

- *University of California, Berkeley*. This project studies technology for intelligent access to massive, distributed collections.
- *University of California, Santa Barbara*. The Alexandria project develops a digital library that provides access to large and diverse collections of maps, images and other spatially indexed information.
- *Carnegie-Mellon University*. The digital video project at Carnegie-Mellon University explores how multimedia digital libraries can be established and used.
- *University of Illinois at Urbana-Champaign*. This project has its focus on information infrastructure and on semantic search of technical documents on the World Wide Web.
- *University of Michigan*. This project is concerned with the creation and evaluation of an agent-based architecture for digital libraries.
- *Stanford University*. The digital library project at Stanford studies interoperability and approaches through distributed object technologies to enable uniform, easy access to dispersed network sources and collections.

Appendix A gives a somewhat more extensive description of the projects performed at the different universities. The Digital Libraries Initiative Phase Two (DLI, 1998b) is a continuation of the joint initiative from the first phase, with funding additions of: National Library of Medicine (NLM, 1998); Library of Congress (LOC, 1998); and National Endowment for the Humanities (NEH, 1998). This project phase is aiming to: supply leadership in research towards a new generation of digital libraries; improve the use and usability of distributed networked information resources; and encourage the concerned communities on new applications regarding digital library technology.

Digital libraries can be seen as serving as intellectual infrastructure, hence this initiative wants to encourage partnering dealings in order to create next-generation operational systems in areas like: education; engineering and design; earth and space sciences; bio-sciences; geography; economics; and the arts and humanities (DLI, 1998b).

2.3 SICS Digital Library Project

As mentioned in chapter 1, the SICS digital library project (DigLib) is run by the Information and Language Engineering group (ILE) at SICS. The project started in the beginning of 1998 and will run until the year 2001. It is funded by SITI (Svenska IT-institutet (SITI, 1998)), and includes several important aspects concerning digital libraries. Some of the issues the project studies are

- digital library infrastructure (Building document collections, as well as constructing a platform with working tools, and digitalisation issues, such as Optical Character Recognition (section below) correction experiments.)
- text and speech analysis tools (Developing new types of analysis algorithms and tools for text and speech collections. The tools will be deployed and tested in the example library. Examples of tools are summarisation and information extraction facilities.)
- information seeking strategies and evaluation. (Studying users' varying information needs and behaviour, and exploring the use of adaptive interfaces.)
- multilingual information access (Including cross-linguistic information retrieval and machine translation.).

In comparison to the DLI project described above, SICS DigLib is a small-scale project. Its starting point is the traditional library and it has its emphasis on the tasks and the users: investigating why people go to libraries and what they expect to achieve there. One issue in the project is the multi-linguality. Almost all research in the area of text analysis is performed on English texts and speech, an important contribution of the project will therefore be tools developed for the Swedish language. This thesis is concerned with both the digital library infrastructure as well as text analysis tools.

Optical Character Recognition

The documents that reside in digital libraries are often originally in paper form; this is also the case for SICS DigLib collection. This means that the documents have to be transformed to digital media, which is done by using a scanner. When using a scanner (or digital cameras or fax modems) an image of the page is scanned into the computer memory as patterns of bits. To enable processing of the text, the individual characters must be recognised and converted into ASCII text code (or the equivalent). This is the mission that the *Optical Character Recognition* (OCR) software must undertake. The techniques used to accomplish this include: segmentation of the scanned page into pictures, blocks of texts and individual characters; expert system technology for identifying the underlying rules that discriminate character; context sensitive modules for the identification of ambiguous letters; learning by examples and by feedback from human trainers (Beekman, 1999).

2.4 Information Retrieval

Since the DigLib Platform is to be used for, amongst other things, testing and evaluating different kinds of information retrieval tools (for text analysis, clustering, etc.) a short introduction to the Information Retrieval area is given below.

About Information Retrieval

The subject area of Information Retrieval (IR) is related to the fields of library science; information organisation and classification; and computer processing (Salton 1989), and the aim is to get accurate access to an extensive amount of information.

Although Information Retrieval as a research area has been a vastly growing object of interest during the past decade (or so) the field started to be explored as early as the 1940s. Retrieval of relevant information from a collection of texts is mainly dependent on how well the document's content has been characterised. For a computer application to 'read and understand' text, it must perform both syntactic and semantic extraction of information. This information, in turn, is to be compared to a query made by the user. The aim of all IR systems is to be able to retrieve all relevant documents and preferably no non-relevant ones. In order to enable a computer to perform automatic indexing (the making of a representation of the text, e.g. a list of words, see section below) and retrieval, the relevance decisions are most often quantified (van Rijsbergen, 1979). Luhn (1959) first defined the field of Information Retrieval, and made some of the first important computational assumptions, such that index terms automatically can be gathered from texts, on the basis of their occurrence statistics. Very frequent terms, and single mentioned terms are furthermore similarly uninteresting in terms of topical modelling of texts. Luhn's work still constitutes the foundation of many IR systems today.

Automatic Indexing

The perhaps most common application of Information Retrieval techniques can be found in the field of automatic indexing. This is used in search and retrieval tools of various kinds and the purpose is to automatically make internal representations of the texts, not to extract keywords to present to the user. Salton (1989) proposes a blueprint for the generation of indexing representations (i.e. terms) from document texts. The blueprint can be viewed as the most basic form of indexing: generating single-terms as opposed to synonym lists and tables of term relationships, the latter according to Salton (1989) being outperformed by the more simplistic formula.

Salton's Blueprint for Automatic Indexing

1. identify the individual words occurring in the document
2. delete common function words by comparing to a stop list (containing high frequency words like: and, or, the, etc.)
3. remove suffixes in order to obtain the word stem
4. compute the term discrimination value, i.e. the weighting factor[3] for each term
5. represent each document in the collection by the extracted word stems, accompanied with corresponding weighting factors
6. when weighting factors for the documents as well as the query terms have been obtained, similarity values can be computed
7. match the query to the documents

This list can be increased by more steps if a thesaurus or word-grouping tool is available. The system for automatic indexing outlined above can be used on document texts as well as queries, assuming that the queries are available in natural language[4]. For more accurate retrieval of documents relevant to the query, additional calculations of term-relevance factors can be made.

[3] Composing the term frequency and the inverse document frequency of the term.
[4] Some adjustments on the calculation of queries are, however, needed to obtain accuracy.

3 DigLib

This chapter gives a presentation of SICS digital library, i.e., a large part of our work. The first section describes the content and the structure of the DigLib collection. The following section describes the deployed search engine (Dienst), including a description of the format for the bibliographic records required by Dienst. Next is a thorough account of the development of KEF (Keyword Extraction Function). The concluding section presents the other facilities added to the digital library.

3.1 The DigLib Collection

Most of the DigLib collection was collected and scanned during the summer of 1998 under our supervision. It consists of language technology and computational linguistics reports and journal articles from the Nordic countries: Nordic Conference of Computational Linguistics (NoDaLiDa); Nordic Conference on Text Comprehension in Man and Machine (NordText); Centrum för datorlingvistik, Uppsala University; Statistical Methods In Linguistics/SMIL Journal of Linguistic Calculus (SMIL); and KVAL Institute for Information Science (KVAL) including papers from International Conference on Computational Linguistics (Coling). SICS has offered to host an archive for reports and dissertations in computational linguistics from Nordic research institutes. This will increase the size of the library, as well as its diversity.

The material ranges over a relatively large time span: from 1969 until today, and most of the material has not been digitally available before. The languages represented in the collection are English, Swedish, Danish, Norwegian, French, and German. The main part of the collection is, however, written in English. The older documents have been scanned, while some of the more recent material was written for digital media. As the scanner output after the OCR much depends on the quality of the material (e.g., the size of the letters, the printing, and the quality of the paper), the outcome of some texts is barely readable.

The Defined Directory Structure

As mentioned in chapter 1, the desire was to have the collection structured in an intuitive manner for a human administrator, and the documents were collected in accordance with their origin. In Appendix B a schematic drawing of the hierarchical structure of the collection is given.

A large amount of the documents were not originally written for digital media, and have, as mentioned above, been scanned. To enable a user to view the original layout, each scanned page is saved in a Tagged Image File Format (TIFF, 1992). Each document can have multiple formats, where each format is associated with one separate file, or set of files. A paged format such as TIFF has a directory with the name *tiff*, in which the TIFF pictures for a document are collected. As there are many pictures to each document it should be convenient for the human administrator to have them collected in a single place, rather than scattered around. All different formats, associated with an individual document, are collected in one directory. Most of the documents were made available in HTML (HyperText Markup Language (HTML, 1998)) since they were to be accessible through the World Wide Web (Berners-Lee, Calliau & Groff, 1992). Other formats are plain text, Word format and RTF (Rich Text Format) (RTF, 1997).

One important aspect for us to consider was the naming procedure. Each file should be given a unique name that reflects the origin of the document. The same name is given to the directory associated with each document. For more details on this issue, see Nylander (1998).

3.2 Dienst

Dienst (Distributed Interactive Extensible Network Server for Techreports) is a protocol and implementation for distributed digital library servers on the World Wide Web (Davis & Lagoze, 1994). It was developed by Cornell University (Cornell, 1998) and Xerox (Xerox, 1998), and forms the basis of the Networked Computer Science Technical Report Library (NCSTRL) (Dienst, 1995a). Dienst provides four digital library services: storing and delivering multi-format documents (repository services); indexing the documents' meta-information and supplying search engines over these indexes (index services); providing information for interoperability among servers (meta or contact services); and finally, supplying a human front-end to the three other services (user interface services) accessible from a WWW client.

A collection of documents can be divided into two or more subcollections, which, within the Dienst system, are called *local authorities*. The Dienst system provides two kinds of default interfaces: one simple search and one fielded search, between which the user can choose. The fielded search contains four different fields: title, author, abstract, and document id. Dienst combines the search terms in the different fields using the Boolean operators AND or OR; which to use is up to the user to decide. The user must also specify whether the search is to be performed on the whole collection or on one or more specific subcollection/s. The simple search contains one single field, and the search is performed on all indexed bibliographic information. (Dienst, 1995b.) When a search has been performed, a list of the titles and authors for all matching documents is displayed. By choosing one of the documents, the abstract as well as the different storage formats for the specific document is presented. Depending on format, the user can either view or download the document. Although the search is performed on the meta-information, the whole documents are retrieved.

By using this function in the digital library, a user is able to find a document that she/he already knows of. We called it *retrieve known document* in DigLib.

RFC 1807

The meta-information that is indexed by Dienst is found in bibliographic records - one for each document in the collection. The bibliographic records are in the Dienst documentation called *bib-files*, and their format is based on RFC 1807 (RFC 1807, 1995).

The RFC 1807 defines a format for bibliographic records describing technical reports. It is a tagged format with self-explaining alphabetic tags. It contains about 30 different tags, some of which are mandatory for a bibliographic record following this format. Below is an example of a bib-file, taken from DigLib.

Some tags described by RFC 1807 are mandatory in order for Dienst to perform its search, the others can be added if desired. Additional tags, proposed by the standard, were carefully reviewed on behalf of other tools in our platform, but none was considered necessary. This was partly due to the sufficiency of the tags used for Dienst, and partly as a precaution in case the new version of Dienst, soon to be launched, will demand different structures of the bib-files.

```
BIB-VERSION:: CS-TR-v2.1
ID:: dienstnodalida//NODA93-13[5]
HANDLE:: dienstnodalida/NODA93-13
ENTRY:: August 18, 1998
TITLE:: Clustering Sentences - Making Sense of Synonymous Sentences
AUTHOR:: Jussi Karlgren
AUTHOR:: Björn Gambäck
AUTHOR:: Christer Samuelsson
LANGUAGE:: eng
ABSTRACT:: Clustering Sentences - Making Sense of Synonymous Sentences
DATE:: September 1, 1993
END:: dienstnodalida//NODA93-13
```

The Deployment of Dienst

Although the Dienst system is developed for distributed collections, the Dienst implementation for DigLib is used as a standalone server, not connected to other Dienst servers. The deployed Dienst system is version 4-1-9, and the deployment comprised two of Dienst's four library services: the index services; and the user interface services.

The Dienst system was tailored and installed, and changes were made to the provided interfaces. As several of the formats present in the DigLib collection were undefined, this had to be done by modifying the concerned scripts.

The work with Dienst was in itself not difficult, per se. It did, however, require quite some time and effort, an experience we share with Baldachi, Biagonini, Carlesi, Castelli & Peters (1998).

The Dienst Directory

Dienst requires a specific database structure since it is unable to search more than a total of two hierarchical levels. To accomplish this structure, a specific Dienst directory was set up (*bibliotek/DienstBibliotek*) with links to the actual directories. This directory contains one directory for each local authority, where the links are collected (Appendix B).

3.3 The Keyword Extraction Function

In this section, we will describe the theoretical basis for the *Keyword Extraction Function* (KEF), a tool developed and implemented by the authors. We will also give a description of Swecg, the tagger used by KEF.

Most automatic indexing algorithms are developed for making internal representations of the texts, and not to extract information to present to the user. Hence there is a need to look closer into the matter of automatic keyword extraction, having the user in mind. One of the interactions proposed by Belkin and Carballo (1998) is "determining the content/structure of a set or a collection of information objects". This can be met by visualising the database in a textual manner, by for example presenting keywords in a way that resembles an index in a book – something most of us are familiar with.

Our aim when developing KEF was to make a function that excerpts all relevant words from the collection and presents them in the way described above. KEF is developed for Swedish texts and the option is called *index words* in DigLib. (For an example of the output from KEF, see Figure 3.1.) The primary purpose of KEF is that a user can, by choosing a word, find the documents that

[5] The ID corresponds to *local authority//document name*.

contain the desired word. The user can furthermore influence the amount of retrieved documents by stating the desired number of occurrences for a chosen word. This function is described further below. By presenting the content as an index also fills another function. Namely, when using search engines, users attempt to specify their information need in words or clauses, i.e., formulate a query. People generally find it more difficult to recall words or facts from memory than to recognise the words or facts in question when seeing them (Andersson, 1995). A user can thus look at the words in this list for inspiration when writing queries.

3.3.1 The Swecg Tagger

A tagger marks every word in a sentence in regard to its word-class and assigns phrasal constituents to them. The tagger used for KEF is called *Swecg* (Swedish Constraint Grammar), and originates from the University of Helsinki (Birn, 1995). Swecg also constitutes the base for the Swedish grammar *Svensk* (Olsson, Gambäck & Eriksson, 1998). Swecg comprises three modules: morphological analysis (SWETWOL); disambiguation of morphological ambiguities; and assignment of syntactic function. The first module, the morphological analysis, finds the base-form for each word in a sentence and assigns word-class. When several interpretations are possible, all alternatives are given. If there is only one syntactic function possible for the word, the syntactic function is assigned at this stage of the analysis. The second module picks the contextually correct interpretation for words that have been given two or more word class tags in the first stage. Finally, the third module assigns syntactic function labels, optimally one to each word. This third stage is optional, i.e., the final output can be with no or little syntactic analysis (derived from the first stage).

Words absent in the lexicon are marked *<NON-SWETWOL>* (mostly proper names). Compounded words are divided by underscore (_). An asterisk (*) indicates capital letter: in names; in abbreviations; and in beginning of sentences. The syntactic function is preceded by an @. The remaining tags specify the word-class as well as the meaning of the grammatical inflections of the words (e.g., specifying that a certain noun is N-gender (*NEU*), or that a pronoun is possessive, first person plural (*POSS-PL1*)). However, the only tag relevant for KEF is the noun tag (*N*).

Below you will find an example of an analysis by Swecg made without the third module (without assignment of syntactic functions) of the sentence: *Plattformsbyggandet till vårt digitala bibliotek har gjorts med sicsDAIS*.

```
"<plattformsbyggandet>"
"*plattforms_bygga" DER/-nde N NEU DEF SG NOM
"<till>"
"till" PREP
"<vårt>"
"vår" <POSS-PL1> <MD> DET NEU DEF SG NOM @DN>
"<digitala>"
"digital" A UTR/NEU DEF SG NOM
"<bibliotek>"
"bibliotek" N NEU INDEF SG/PL NOM
"<har>"
"ha" <AUX> <SUPINE> V ACT PRES
"<gjorts>"
"göra" V PASS SUPINE
"<med>"
"med" PREP
"<sics*d*a*i*s>"
"sics*d*a*i*s" <NON-SWETWOL> N SG NOM <?>
"<$.>"
"$." CLB <PUNCT>
```

Output from KEF:

- plattformsbygga
- bibliotek
- sicsdais

Figure 3.1. The text to the left, run through KEF, would generate the following keywords.

3.3.2 The KEF Algorithm

When examining which words to choose as index words, the starting-point was a reflection by Källgren (1984) that the nouns seem to carry more information than other word-classes in a text, and that these most often are used in IR queries (Källgren 1984; Källgren 1992). We initially considered having single nouns as well as whole noun phrases (e.g., *the green house on the hill*) as index words. Analyses of five arbitrary but representative documents in Swedish from the collection did, however, not indicate that whole phrases more exhaustively represented the content than the single nouns. In addition, Ingwersen (1992) states that automatic indexing techniques based on the single words occurring in texts are quite effective.

In KEF we prefer high coverage to high quality. The reason for this is that KEF is an experimental algorithm; it ought to be wiser to start with an algorithm that excerpts too much information and from this point refine it to exclude non-relevant words, than to take the risk of missing some relevant words entirely. Some restrictions are made, though; not every word classified as a noun will be presented. A noun occurring only once in a document will be disregarded, in accordance to Luhn (1959) and Salton (1989) stating that terms with a single occurrence are uninteresting when indexing a text. Another delimitation is removing all words containing strange characters. Due to shortcomings of the Optical Character Recognition (OCR), the output often contains strange characters, such as $, #, numbers, etc. In order to reduce the amount of non-relevant words still remaining after taking these measures, we pondered whether the word length could be used as a selection criterion. After a thorough examination of the KEF output, we decided to remove all words consisting of three letters or less, since more than 90 percent of these did not have any substantial content This approach may however be less suitable for documents that are not scanned. (see Appendix C).

One possibility to further reduce the number of non-relevant words, could have been to remove nouns that Swecg had marked as *non-swetwol*. This would have meant omitting a great deal of words that, due to scanning errors, are nonsense words (although accepted by Swecg), as well as many words in other languages than Swedish (some documents contain abstracts written in English). However, if omitting these there would have been a risk of missing relevant terms, as many documents in the collection contain terms that are too specific for Swecg to recognise. After the second stage of Swecg, some words still have more than one word-class assignment. In that case KEF will consider the word a noun, i.e., it will be included in the index list. Both these decisions are supported on our standpoint to prefer high coverage to high quality.

The asterisks added by Swecg to mark capital letters, are removed when the nouns are extracted. Names of people and places are, in the output from KEF, in lower-case, since no restoration regarding capital letters is performed. Although words that are recognised as proper names by Swecg are marked as such, some pass unnoticed. Sometimes words that are not proper names are interpreted as such. This means that it would be hard to make confident discernments regarding the different occurrences of capital letters, and we therefore chose not to look further into the issue.

Although the algorithm to some extent takes the shortcomings of the OCR into account, some errors will remain in the final output of KEF. The algorithm is, however, appealing because of its simplicity.

3.3.3 The KEF Modules

The main Keyword Extraction Function program is a standalone Java Application (Sun, 1998). Its construction follows, to some extent, the structure outlined in the blueprint by Salton (1989),

section 2.4, *Automatic Indexing*. Some features in the blueprint were, however, not relevant for KEF, such as: suffix stripping; computing weight values; and matching between a query and document.

The part of KEF described above, is only run when a new Swedish document is added to the digital library (by adding a link to a link file, described below). Because of this, no effort was made in creating a speedy KEF. The second part is a content handler. That is, the visible part of the underlying program made to be used in sicsDAIS (see section 4.2).

The first step performed by KEF is to access the documents that are to be manipulated. All links to the Swedish documents are put in a file (*bibliotek/linkSwe/links.txt*), and KEF is able to read documents in HTML as well as various text formats (Word documents, plain text, etc.). This step is followed by an identification of word-class for each individual word in the texts; for this the Swecg tagger is used. Swecg is unable to handle hyphenation at linefeed, as well as some other (strange) characters. Thus one step in KEF is to remove any hyphen found at the end of a line and to pull the two word parts together. Remaining modules of KEF extract appropriate terms from the tagger output; computes statistical features, i.e., term frequencies; and compiles the result.

Development of the modules and refinement of the algorithm was carried out in parallel. Analyses were also made in-between the development of different modules, in an attempt to reach correctness of the algorithm.

For a more general overview of KEF, see Figure 3.2. The modules are, in more detail, outlined as follows:

- access and read the list with URL links to Swedish documents
  for each URL in the list
    - access document
    - check extension
    - remove HTML tags (when html extension)
    - remove hyphens
    - send to Swecg for tagging of document
    - identify nouns in Swecg output
      for each noun
        - remove noun with strange character/s
        - remove noun shorter than four letters
        - remove underscore
        - compute occurrence
        - remove noun with occurrence one
    - sort remaining nouns alphabetically
- collect all words with common initial letter, and
- write all words to different files regarding initial letter.

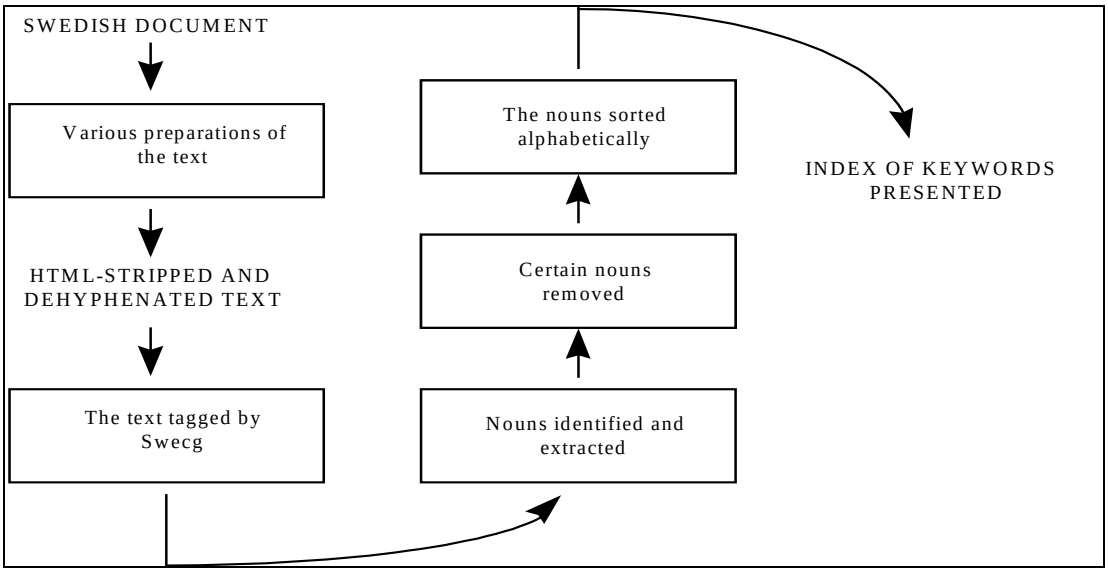


Figure 3.2. A schematic overview of the Keyword Extraction Function.

3.3.4 The Visible Part of KEF

When choosing the *index words* option in DigLib, the interface of KEF is shown (see Figure 3.3). The user can select desired letter by clicking the corresponding button whereupon the index words, beginning with the chosen letter, are presented alphabetically. The words are presented in a clickable list. When the user has selected a word from the list, the titles and the authors of documents containing the word are displayed. In addition the number of documents in which the word occurs, and the minimum number of occurrence, the maximum number of occurrence are presented. Figure 3.3 shows, that there are 9 documents containing the keyword *sida*, and the occurrence of the word in each document ranges between 2 and 38.

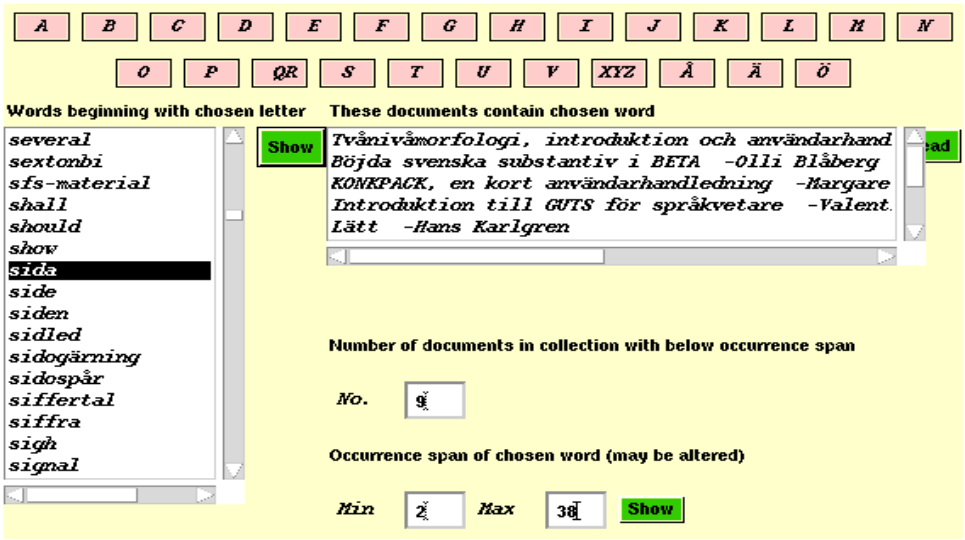


Figure 3.3. The KEF display when a keyword has been chosen.

At this stage the user has the possibility to influence the number of retrieved documents by altering the occurrence span. Figure 3.4 shows that by stating a span of 4 to 7 occurrences, the number of documents reduces from 9 to 2.

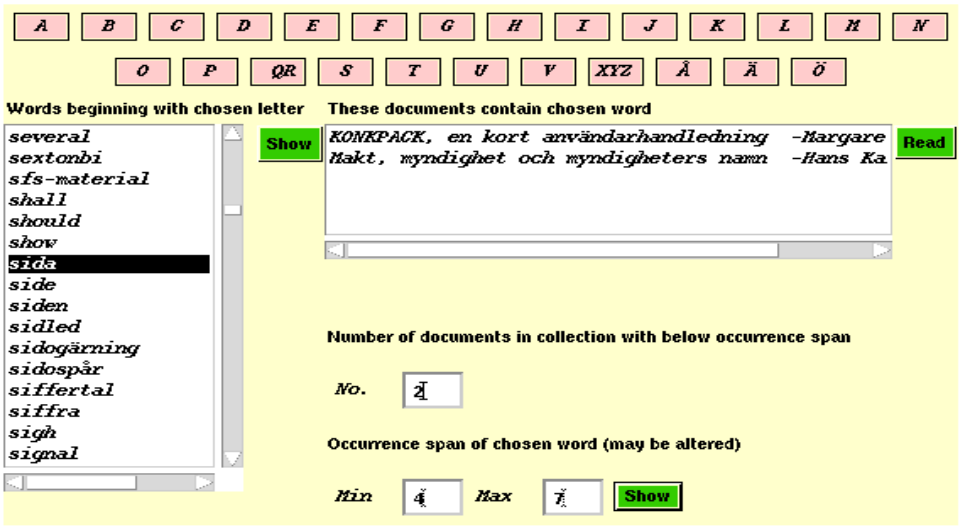


Figure 3.4. The KEF display when the occurrence span has been altered.

By experimenting with occurrence spans, a user can get indications of which occurrence span for a word has a good discrimination ability, regarding both individual documents and the frequency of the word in the whole collection.

3.3.5 Suggestions about KEF Evaluations

Although based on a comparatively simple algorithm, KEF, in our opinion performs quite well, i.e., the index mirrors the content, even if the word presentation hinders an instant overview. Nevertheless, it provides an easy way to find documents containing a certain word, and it contributes by supplying a long-felt want in the area of Information Retrieval (ERCIM Workshop, 1998). An evaluation of KEF could probably be carried out in several ways, and we will in this section give some suggestions about the issue.

In order to study to what extent the index words mirrors the content of the collection, a suitable number of test subjects could be asked to look at the index and, in one way or another, describe what kind of documents they would expect to find there (i.e., what subjects they treat).

The evaluation of the other aspect of KEF, i.e., to find a document containing a certain word, could be performed in two ways:

1. The test leader assigns one or more subject word/s, to each of a number of documents. The test persons are thereafter asked to find documents about that topic.
2. The test person is asked to choose any word in the index, and thereafter read one of the retrieved document. S/he is after the reading supposed to tell to what extent the expectations regarding the content were fulfilled.

3.4 Other Functions

Apart from the search engine and KEF, we added a few more options to the library. One is called *about* and gives information about the DigLib platform. Another option, *help desk*, describes the digital library, i.e., the collection and the different functions.

A study on interface design for IR interactions (Hansen, 1997), states that the following should be provided to enhance user satisfaction: a database collection description including a specification of the time coverage and a presentation of the formats present; a keyword list; and information about database updates, including date for last change and which papers have been added. The two first requests were already provided for (by *help desk* and *index words*). To provide for the third, we included the function *new documents*, where a human administrator can add information about new documents in the collection. The titles are hyperlinked to the actual files to enable immediate viewing by the user, if so desired.

Apart from having access to various tools for searching a digital library, there should also be a function which does not involve putting the information need into an explicit query (Croft, 1993). This is in case the user either does not exactly know what she or he is looking for, or how to formulate her or his information needs. We therefore added the option *browse the collection*. This information seeking interaction is by Belkin & Carballo (1998) referred to as ”recognising useful information by scanning through an information resource”.

4 Building the Digital Library Platform

This chapter describes the concluding part of our work, i.e., the integration of the different parts, described in chapter 3, in one interface. The chapter gives a general description of the system used for implementing the platform (for those interested in the more rigid details of the sicsDAIS system, see Espinoza (1998)); the construction of the platform itself; and the design of the graphical user interface.

4.1 SICS Dynamic Agent Interaction System

As discussed in chapter 1, the initial proposed solution for gathering and accessing several tools in the digital library was through Web pages. The platform, however, required flexibility so that adding or removing tools would not involve changes in remaining modules. In using the Dynamic Agent Interaction System (sicsDAIS) we could accomplish this flexibility. The main idea of sicsDAIS is to provide a common interface enabling users to interact with multiple agents. It is built in a way so that new agents can be introduced without informing the system in advance (agents can even be added during runtime), and they can just as easily be removed. The system is open: it is possible to introduce new services for the user to access without modifying the existing interface system or services. This means there are no Web pages or CGI-scripts to change. The system is dynamic: services can come and go during runtime. Due to the nature of the sicsDAIS system, a new service which provides its own interface may be plugged into the set of services the user is already accessing. The system was developed at SICS for an Irish project called KIMSAC[6] (Charlton et al., 1997).

[6] KIMSAC: Kiosk-based Integrated Multimedia Service Access for Citizens

What is sicsDAIS?

sicsDAIS is an interaction system constructed to enable several agents to present themselves as well as their services to a user or to one another. To make use of the system one creates *content handlers*, which serve as a transition spot between the agent and the interaction system. In this way the agent's innate behaviour and interface can be preserved if desired - the actual agent resides outside of the system and is reached through its content handler. sicsDAIS provides means for: layout; communication between content handlers; and the interaction with the user.

When having a number of content handlers, which each corresponds to one or more agents (or other types of programs, for that matter), there is a need to arrange them in a suitable way for the user. For the sicsDAIS to know how to go about doing this, Presentation/Interaction (P/I)-descriptions must be made. These are scripts, or declarative descriptions (or commands) of how the content handlers are to be configured before creation and display. The script format is a language reminiscent of both Lisp (Lisp, 1996) and KIF (Knowledge Interchange Format) (Genesereth & Fikes, 1992). They contain information about which content handlers to display; their preferred layout; their individual properties (size, colour, etc.); their interdependencies; etc.

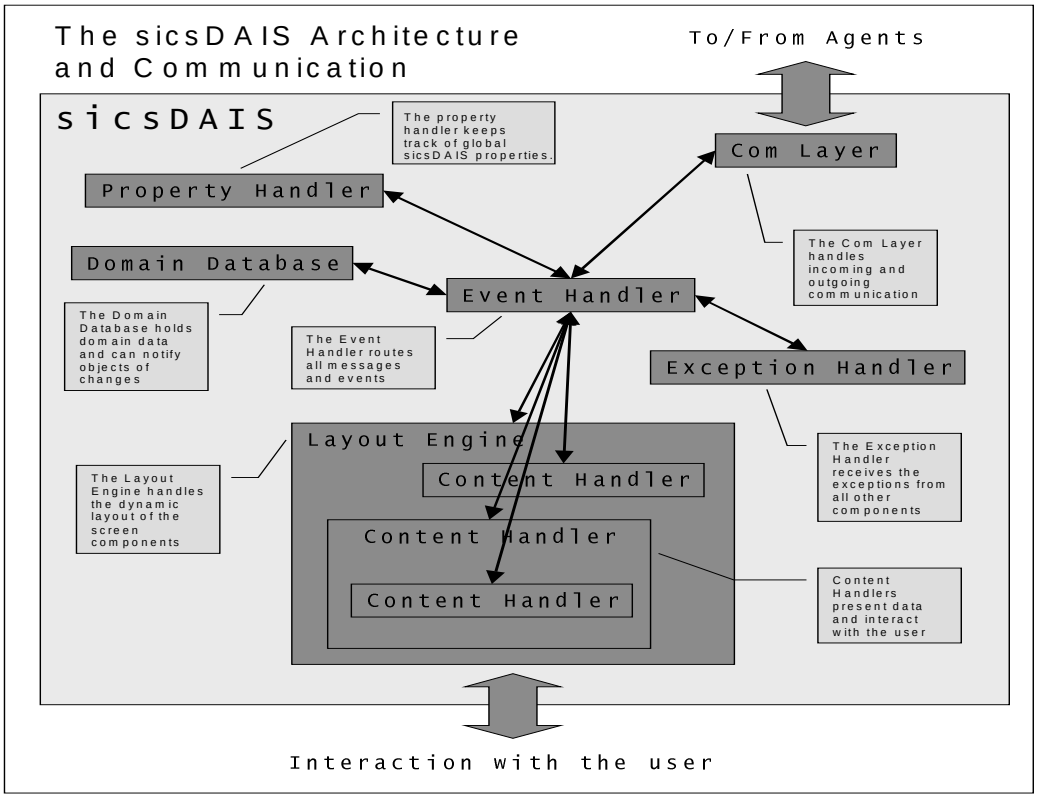


Figure 4.1. The sicsDAIS architecture, with its different components. From Espinoza (1998).

Figure 4.1 shows the components involved in sicsDAIS. These are

- the Com Layer (handling the communication with the agents)
- the Layout Engine (managing the layout of the content handlers in the presentation)
- the Event Handler (for messaging between components)
- Content Handlers (mini representations of the agents)

- the Domain Object Database (for storage of shared data)
- the Exception Handler (for error and exception handling)
- the Property Handler (managing the client side's instance of sicsDAIS).

A general description of how the system works will be given in the following sections, focusing on the components most highly topical for the creation of the DigLib platform.

Event Handling

The event handling component can be seen as the heart of sicsDAIS. All messages passed between the different modules go here, get appropriately processed, and are shipped off to the receiver. For this to work properly, all components' unique IDs get registered in the event handler when created, to secure the message passing accuracy. A message is a script with a certain content. When messages go between components within the system that is all there is to it, but as soon as the message comes to or from an agent (that is outside the sicsDAIS), the script is packaged in another message format, containing information about the sender and receiver, what language the script-part is in, and so on. For the KIMSAC project this was KQML (Knowledge Query and Manipulation Language) (KQML, 1998). For the DigLib project, a CGI (Common Gateway Interface) (CGI, 1998) compatible message format was used.

Content Handlers

The visible part of an agent (or some other functionality) in the sicsDAIS system is represented by a content handler. A content handler is a Java class whose object is created when it is loaded into sicsDAIS, according to pre-stated layout preferences to the layout engine.

To create a content handler one must sub-class either the *AtomicCH* class or the *CompositeCH* class, which both are provided by sicsDAIS. These classes incorporate the methods needed for the content handlers to co-operate with the system as a whole (both within sicsDAIS as well as out towards the agents). Apart from this, the creating of a content handler is much like writing a Java Applet (Flanagan, 1997).

An atomic content handler (when sub-classed and instantiated in sicsDAIS) represents one unit in the user interface, like a button or more complex combinations of AWT (Abstract Window Toolkit) components (or the like). Messages sent to an atomic can therefore not be received by any isolated part within, but go to the atomic as a whole. What if there is a need for several, say buttons in the user interface, practically identical apart from, say different labels? Then one can, instead of coding an atomic with a complex interface and event handling for a fixed number of buttons, write only one atomic content handler. This atomic can for example contain the desired button and a method for setting the desired label (an event that is trigged from the P/I-description (script) and mapped to the method in the content handler as the content handler is being created). This means that the number of displayed buttons, as well as their individual labels, are stated in the script and the changing of the number of buttons would not involve any re-coding, nor compilation, of the content handler class.

Collecting several content handlers in this manner, through descriptions in a script, forms a composite presentation in sicsDAIS. If one should want a somewhat higher functionality one should however sub-class the CompositeCH class. In doing that, several content handlers can reside in the same class, methods for layout can be altered or added, etc. However functional, this approach looses somewhat in flexibility.

If one would erase a description of a content handler from the script, and call that script anew, it would in essence mean that you have lost contact with the agent it represents. Hence the flexible feature of adding and removing during runtime.

The Layout Engine

If, as stated above, the event handler is the heart of sicsDAIS, the layout engine can be viewed as the genetic code (although easy to manipulate). That is, this component takes care of the presentation of the content handlers. The layout engine is in itself a content handler, albeit on root level. This means that it is responsible for handling the adding and the alteration of behaviours of all main composite (and atomics on root-level) content handler assemblies defined in the script.

A message can be sent to the layout engine, for example telling it to rearrange the display of the atomic content handlers residing in one particular composite, from, say vertical layout to horizontal.

4.2 Using sicsDAIS

The original Communication Layer in sicsDAIS had three kinds of interfaces for communication with the agents: a socket interface (for connecting via a socket); a file interface (mostly used for testing sicsDAIS towards P/I-descriptions etc. stored in files); and the Agent Services Layer (ASL) interface (a communication architecture for agent communication). To be suitable for the DigLib platform, an additional interface had to be added to the system, namely the HTTP (Hypertext Transport Protocol) (RFC 2068, 1997) one. Along with this came suitable methods for handling the scripting (i.e. the CGI message format mentioned above), and a content handler for presenting HTML, all kindly provided by the developer of the interaction system. sicsDAIS as such, also had to be transformed, from being a standalone Java Application to becoming a Java Applet. This to allow the digital library platform to be accessible on the WWW.

The Content Handlers

For the DigLib platform, nine different content handlers were constructed: one for the six options (see section 3.2, 3.3 and 3.4), two shared content handlers for the tools to be able to show the results in HTML, and the rest containing buttons and events etc. for the GUI. A complete list of the content handlers that were written for the DigLib platform are found in Appendix D.

The simplest content handler of them all, where functionality is concerned, is the *CHPicture* content handler. This class is able to load one of five different images upon creation, depending on what parameter is stated in the script, and display it. One image (Figure 4.2), the *quit-button*, has an event tied to it. When clicked, the digital library closes down.

Figure 4.2. The quit-button.

The *buttonLabel* content handler is very much like the CHPicture class in that one can state what image to display. This class has three different button images to choose from, each actually consisting of two, one showing a clicked and one a not clicked button (Figures 4.3a and 4.3b). This

atomic does, however, have somewhat more advanced features. In the DigLib interface, a row of buttonLabel atomics is displayed to the left. For the individual buttons to communicate with each other, every instance of the content handler subscribes to a value in the domain object database. In this way, every button gets notified as soon as a new value is added, and can check whether the event occurred in themselves. If so, they change image (to clicked if not clicked). Meanwhile in the script, a new value is sent to the database (telling the other buttons to get unclicked), as well as a command to trig the event in the content handler which sends a new script, with a new scenario of content handlers, to be loaded and presented to the user.

Figure 4.3a. An instance of a buttonLabelCH. Button not clicked.

Figure 4.3b. An instance of a buttonLabelCH. Button clicked.

The two content handlers described above do not represent an underlying program. The *dienstCH* (Figure 4.5) and the *kefCH* (Figures 3.3 and 3.4) on the other hand, do. They both have a somewhat more complex user interface, and perform a great deal of action. There are, however, no sicsDAIS-level events tied to corresponding events in the content handlers. dienstCH has access to e.g., the list of local authorities and the indexes produced by Dienst. It sends the user input, entered in the interface, to Dienst (which matches the query), and takes care of the returned result by directing the information so that it can be viewed in the HTML viewer window (described below).

The *kefCH* content handler is the second part of KEF, and it has access to the keyword files produced by the main KEF part (described in section 3.3). In coherence with the user interactions with its interface, kefCH accesses the appropriate file and manipulates its contents (stores information about e.g., titles, authors, URLs and, of course, the keywords in Java objects). It then presents the words (with its corresponding occurrence values) on the screen, and if the user wishes to look up a particular word, it presents the title/s and the author/s that are stored in that particular word's object. If the user wants to view a document, this document's URL is sent to the *webCH* content handler (described below), and finally, the HTML viewer window displays the document.

Saying that the CHPicture content handler class functionally is the simplest is not altogether true. The *emptyCH* content handler class is actually a dummy, who's only task is acting as a placeholder in-between other content handler objects. The content handler classes *infoBox* and *helpCH* contain various methods for getting hold of information from the script, and simple interfaces for presenting this information. The *HTMLViewerCH* class is, however a bit more complicated than that. This class, when instantiated, creates a browser window (by wrapping the IceBrowserLite class (Icesoft, 1998)). It has a horizontal scrollbar, and two buttons enabling the user to go back or forth in a history list of previously viewed pages (HTML or text etc., depending on what function in the digital library that has been used). The HTMLViewerCH resembles an ordinary Web browser window, only with somewhat fewer finesses. The last content handler, *webCH*, works in close co-operation with the HTMLViewerCH. It has methods to: receive URLs stated in a script, or sent from another content handler; access these web pages; and send their contents to the HTMLViewerCH object (which shows the content in its window).

The Scripts

Seven different scripts were written for the DigLib platform. The example below is an extraction from the script that creates the option buttons found on the left in DigLib's interface (Figure 4.4). A complete list of the scripts that were written for the DigLib platform are found in Appendix E.

The script below would, if used in its current shape, create and present two buttonLabel objects (*createAsset*), placed in a vertical row. Under the *setProps*-tag one sets properties or events that are to be performed during the creation of the content handler object. In this example *setButton* "yellowLarge" chooses what image should be loaded for the button display (Figures 4.3a and 4.3b), and the *setLabel* sets optional string label. The *setReactions*-tag tells the content handler what to do if the button is pressed, namely: send a new value to the domain database (*dodbSetValue*); send a script to tell the browser window (HTMLViewerCH) to hide (in case the window was shown in the previous session); then execute the *changeFrame* method which gets a new script at stated URL and displays its contents.

Example of script

```
(sendScript "layEngine"
    '(newAsset
      (createAsset
       (setProps
         (listof
         '(layout "absolute")
         '(expand "none")
         '(alignment "north")
         '(padding 5)
         '(dodbDefProp "button" "id" "")
         '(dodbDefProp "web" "htmlviewerid" "")
         '(dodbDefProp "script" "id" "") ))
         (setParts
            (listof
            '(createAsset (setProps
                  (listof
                  '(layout "vertical")
                  '(expand "none")
                  '(alignment "west")
                  '(preferredLocation 0 20)
                  '(padding 5) ))
               (setParts
                  (listof
                  '(createAsset "buttonLabel" (setProps
                     (listof
                        '(setLabel "Retrieve Known Document")
                        '(setButton "yellowLarge")
                        '(size 209 43)))
                           (setReactions
                           (listof
                           (listof "init4" '(dodbSubscribe "button" "id" "tag"))
                           (listof "buttonPress"
                           '(block
                              (dodbSetValue "button" "id" "Retrieve Known Document")
                              (sendScript "14" '(hideBrowser))
                              (changeFrame"http://kotov.sics.se/dienstScript.lsp"))))))
                  '(createAsset "buttonLabel" (setProps
                     (listof
                        '(setLabel "Index Words")
                        '(setButton "yellowLarge")
                        '(size 209 43)))
                           (setReactions
                           (listof
                              (listof "init4" '(dodbSubscribe "button" "id" "tag"))
                              (listof "buttonPress"
                              '(block
                                 (dodbSetValue "button" "id" "Index Words")
                                  (sendScript "14" '(hideBrowser))(changeFrame
                                  "http://kotov.sics.se/kefScript.lsp")))))))))))
```

4.3 The Design of the Interface

A well-designed interface displays high visibility (how easily and directly the user sees what can be done and how to do it); gives appropriate feedback (how easily and quickly the user can determine the results of actions); has concrete mappings (correspondence between desired function and requested action); and high affordance (indications of how something is used) (Zetie, 1995).

The main focus, when designing the interface for DigLib, was not the design of the different functions; the effort was put into the integration of the tools in one single user interface. For example, only minor changes were made to the result list of Dienst, and the Web pages displaying the links for browsing were not adapted for DigLib.

Guidelines for the Design

Our intention when designing the interface, was to give the user a sense of staying at the same place when performing the different tasks, in order to minimise the feeling of being lost. This was realised by dividing the interface in two, where the area to the left contains a function menu and the other interaction (i.e., the different functions) takes place in the right part of the interface (Figure 4.4). A function is selected by pressing the corresponding button, and the button remains pressed while the function is active. By having the menu of all main functions in the system visible at all times, the user will hopefully not loose the overview and can at any time turn to another function. The vertical line, dividing the two parts of the interface, is not connected to the top nor to the bottom, to retain the feeling of unity.

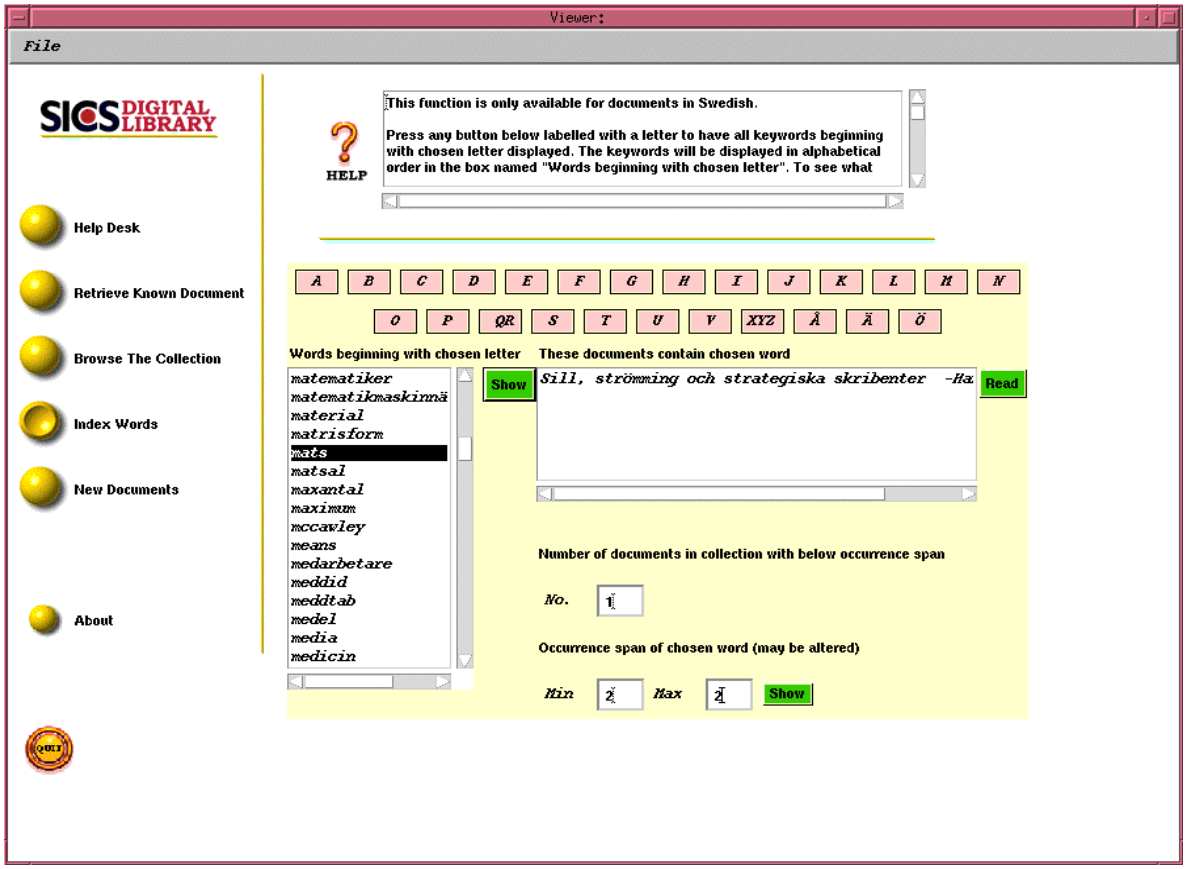


Figure 4.4. DigLib with the working session *index words.*

Our aim was to create an interface for DigLib with a lot of space, as this ought to enhance the overview of both the different options and the work that can be carried out in each specific function. Zetie (1995) states that user performance decreases when there is little space, and recommends at least 50 % of a textual screen to be left empty. The help for each separate function will be displayed only when requested by the user. The help text can furthermore be removed when no longer needed. The help is retrieved in the same way, by clicking the question mark, and is found at the same place for all functions (see Figure 4.4).

Design for Dienst

When adding Dienst to the platform, some design choices were supported by a user evaluation for IR interface design performed by Hansen (1997), where Dienst was the system studied.

Some users in Hansen's study were unaware of the fact that the system actually displays the entire documents (and not only abstracts) for viewing or downloading, therefore this information should be stated more clearly. Better instructions for search syntax was also a request. This was, however, not an issue on our part, since we removed the option to choose what type of search to perform. We made the Boolean AND search prevailing, as searching on all documents that, for example, either contain the word *syntax* in the title OR are written by *Peter*, must be quite rare. In addition, we chose not to include the document id field, as it is unlikely that a user will remember and use these. Nor did we include the simple search field. The reason for this was that Dienst was added to the platform to provide search for known documents. We therefore made the assumption that a user will know whether, for example, the name she/he wants to search for is an author or a name occurring in a title.

Figure 4.5 below shows a Dienst search performed on author *Benny Brodda* in the subcollections *NoDaLiDa* and *SMIL*.

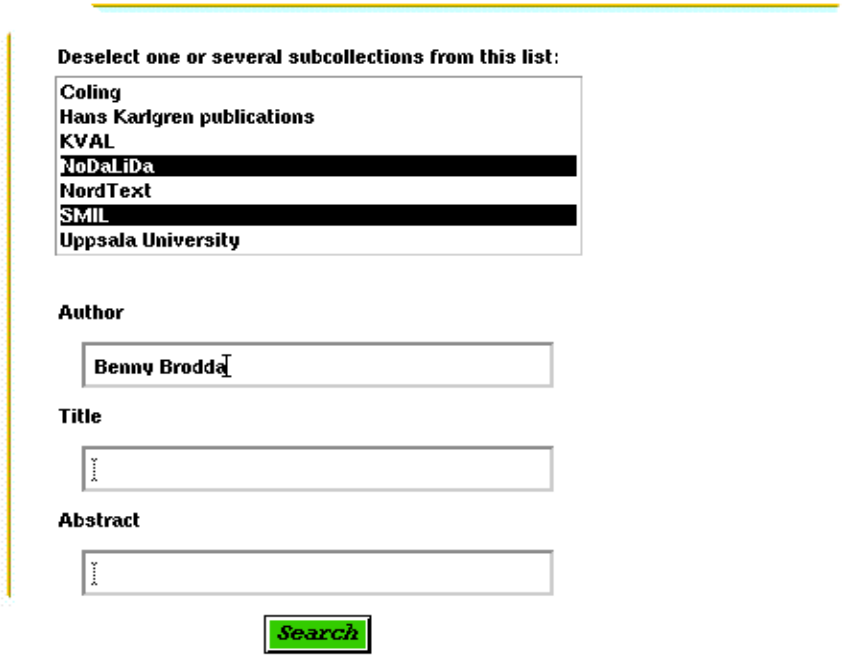


Figure 4.5. The Dienst interface when active in DigLib.

In the interface provided by Dienst, the default setting regarding local authorities is that neither the whole collection nor any subcollections are pre-selected. This is unfortunate since most users, according to Jones, Cunningham & McNab (1998), accept the default settings regardless of what these are. In our design, we therefore chose to have all subcollections pre-selected by default, leaving the user with the possibility to deselect unwanted ones.

When using Dienst, the list of retrieved documents is presented in a separate browser window (Figure 4.6). This enables the user to move the window around (click and drag), and it will be possible to reformulate queries, while at the same time viewing the result of the most recent search. When choosing to view a document, the same window is used for presenting the text. There are two buttons in the window: *back* (so that a user can go back to the result list and choose to view another document) and *forward*.

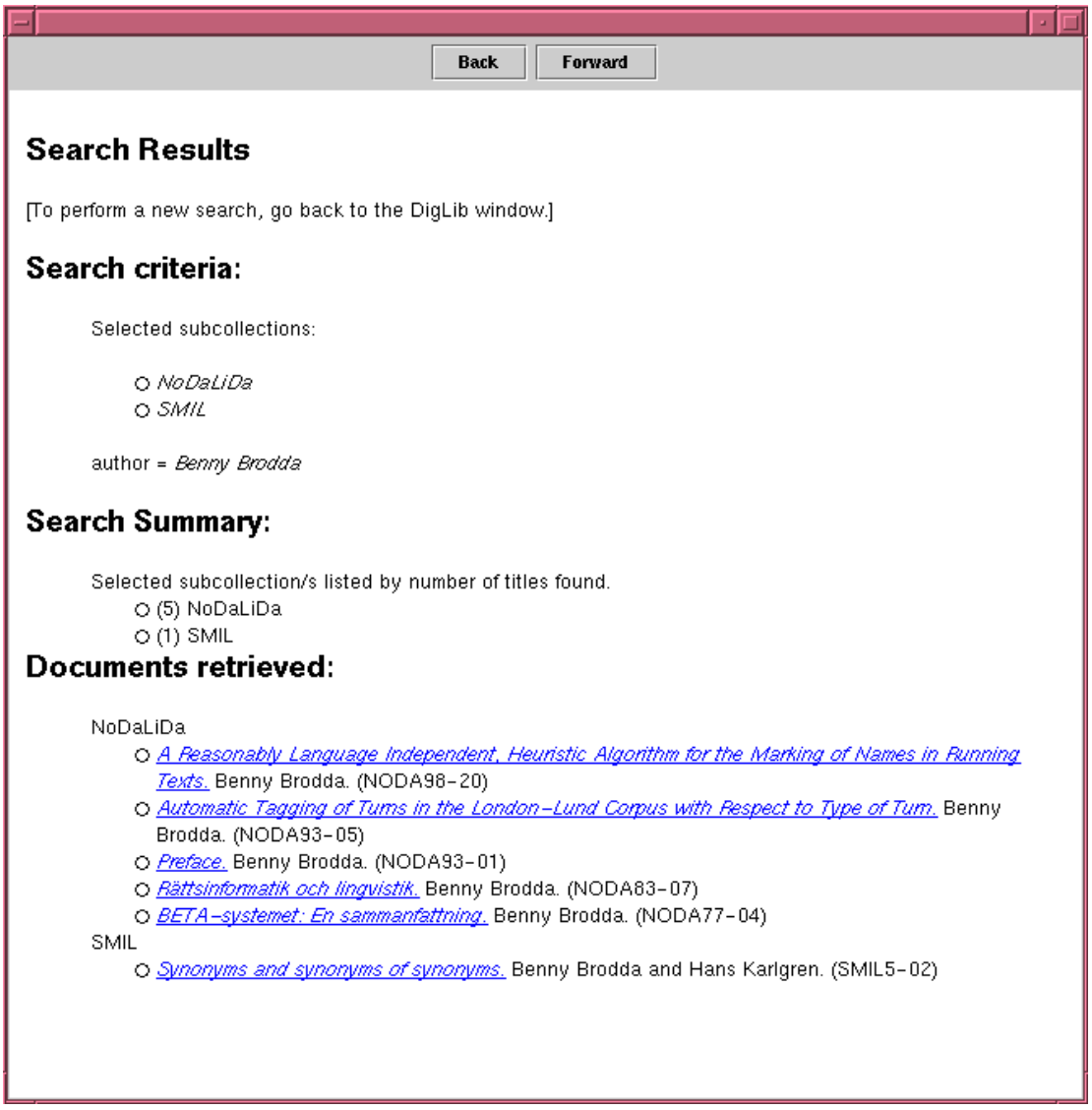


Figure 4.6. Results from the Dienst search performed in Figure 4.5.

5 Future work

In this section we will give some suggestions about future work, most of which were ulterior to our master thesis. The suggestions are more or less necessary to attend to, but would all improve the functionality of DigLib. The subject matters outlined in this chapter are not exhaustive.

5.1 Other Tools

The functions taken into use in the DigLib platform so far are insufficient, and more tools need to be deployed to meet other kinds of user demands. As Dienst performs its search in meta-information, there is a need for a search engine performing a full text search. The system Prise (developed by National Institute of Standards and Technology (NIST, 1998)) could be an appropriate choice. The system requires that the documents are tagged in Standard Generalised Markup Language (SGML, 1998) with accompanying Data Type Definitions (DTD, 1998).

Prise also requires a Z39-50 client. Z39-50 is a standard for information search and retrieval via a network, and is based on the client/server model. It is also an Internet information retrieval protocol (RFC 1729, 1994). There is an already working installation of Prise at SICS, but to add the system to the DigLib, one would need to find a Z39.50 client, preferably as a Java Applet. The ones available on the market (as freeware) could work but would need to be somewhat modified.

We initially looked at a tool, developed at Helsinki University, called WEBSOM (Kohonen, 1997). This tool produces a self-organising map for visualising the content of a set of documents. We did, however, not succeed in our attempts to obtain a licence for the program from the responsible parties in Finland.

5.2 KEF

The first step KEF performs, is locating all Swedish documents in the collection. This is done by accessing the file containing the URLs of the relevant (i.e., Swedish) documents. This file is, at the present, manually maintained by a human administrator, and this workload should be reduced by automating the process. This can be accomplished by implementing language recognition of the documents, which can be done in two ways: As the letters *å, ä,* and *ö* typically occur only in Swedish texts, one can presume that if any of these letters (preferably *å*) is present, the document is written in Swedish. The other alternative would be to automatically look in the bib-records, where the language is specified. When this is done, the file with links to the Swedish documents can be generated.

It should be possible to perform a more sophisticated search within KEF. At the moment, a user can only select one word at a time from the list of index words. It is desirable to enable a user to mark several words, and have a Boolean AND search performed for the marked words.

5.3 sicsDAIS

The sicsDAIS system was originally constructed using Java version 1.1. The AWT (Abstract Window Toolkit) in this version is, however, bug-prone, why sicsDAIS is being changed towards using the Java Swing class (JFC, 1998). In order for DigLib to be compatible (i.e., avoiding errors in layout (see further section 6, *About sicsDAIS*)), one will have to replace the AWT component

declarations, with Swing component declarations. Although this is an important issue, it is not associated with any substantial difficulties.

When a CGI message is sent using sicsDAIS, either from a content handler or from a script, it does take a little while until the result is returned. During this time, a user might start to wonder whether something has gone wrong. One should therefore add some sort of processing feedback to the interface. Perhaps by adding a line containing textual information (e.g., *Processing - please wait*.) somewhere on the screen, or by transforming the mouse pointer into a hourglass. Whatever the solution, it is a necessary feature in an interface.

5.4 Other Issues

The chosen language for interaction is English. A future version of the library should at least provide Swedish as an alternative language; This could be an option in the function menu.

There should be lists of index words for the documents written in the other languages present in the collection - at least in English, which is the most commonly represented language. To accomplish this, one solution could be to run the English version of Swecg, called Engcg (Engcg, 1996), together with KEF. There is, though, a potential risk that Engcg treats compounded nouns as two separate nouns. In that case a word like *computer program*, will be indexed separately as *computer* and *program*. Another alternative could be to use an algorithm proposed by Justeson and Katz (1995), that attempts to identify technical terminology based on repetitive occurrences of, in particular, multi-word noun phrases. This algorithm is implemented at SICS.

In order to establish in what way users interact with the library, and what tasks they perform, transaction logs should be collected and evaluated. I.e., some kind of logging should be added to the system.

It might be helpful for further work with the library to have a proper user study performed of the interface and the tools deployed so far.

In prolongation, other institutions should be able to add their papers and reports directly to the digital library, without having to send them to a human administrator, in turn adds the documents to the digital library. How this could be solved is, however, beyond the scope of this discussion.

6 Discussion

This thesis work made two main contributions: the building of a platform with several functions in a single user interface; and the construction of a tool for keyword extraction (KEF) from Swedish texts. The performed work is rather multi faceted. If compared to the categorisation of digital library research made by Griffin (1997), described in section 2.1, it concerned all three aspects: system-centred research (architecture design - in our case sicsDAIS); collection-centred research (building and providing access to a digital collection); and human-centred research (user needs - adding several tools and the interface). This shows the advantages of small-scale projects such as DigLib; it is possible to get results and, more importantly, to integrate different parts of the project quickly. We have within the scope of this thesis work produced something that most other projects lack, namely a ready-to-use library.

The functions deployed to the platform are *browse the collection*, *retrieve known document*, and *index words*. For the latter function, the developed keyword extraction tool (KEF) is used. In order to come to any conclusions regarding the usefulness of each specific function, evaluations of the same would be necessary. Evaluations concerning the integration of the functions deployed in the library are also important in order to find out whether different user needs can be met this way. We will leave these as open issues for future work.

About sicsDAIS

When deciding to implement the DigLib platform using sicsDAIS, we chose to take on more work than perhaps necessary from a short-term point of view. Writing a number of Web pages would probably have taken far less time, and the user interface would probably have been more aesthetically appealing than the current one is (the physical appearance of the interface was never an important objective in this thesis work, though). On a long-term basis, the choice to use sicsDAIS was, at least in our opinion, better. The use of light-weight content handlers (the actual programs residing elsewhere) created through scripting, helps making the system fast - no heavy executions take place within sicsDAIS - yet, at the same time, the system provides the advantages one would expect of a solid interaction and presentation system.

One problem when implementing the DigLib interface was understanding the details regarding layout properties needed for the scripts (i.e., Presentation/Interaction-descriptions). To place the different content handlers in the right positions according to the design choices, was not straightforward. There are many different property parameters to give values to (e.g., *layout*, *expand*, *alignment* and *padding)*, values that, when altered often lead to unexpected changes of the whole layout. This may, however, be due to shortcomings in the Java AWT, and can perhaps be dealt with by replacing all instances of AWT components with components from the Java Swing class. Another reason could be a mismatch between the component size returned from the content handler class when created, and values stated for the property parameters. Still another explanation can be shortcomings on our behalf, both in understanding the scripting and in writing the content handlers. Since sicsDAIS was developed only recently, and has few practical realisations so far, we can also not eliminate the possibility that the problem resides within the Layout Engine. Most likely, the suggested reasons all contributed equally to the layout problem.

Some troubles have also been encountered when accessing the network through CGI messages (for instance when accessing the Dienst server), but they most certainly depend on network problems rather than sicsDAIS factors. Apart from these and other minor detail problems described above, the platform is, as far as we can determine, robust.

About KEF

KEF provides the user with a keyword index derived from the Swedish documents in the DigLib collection. It would, however, be interesting and probably necessary to develop the interface as well as the algorithm further, e.g., assigning each keyword a value based on its occurrence in each specific document and the length of this document. By performing some kind of user study on KEF, various refinements based on the evaluations could be made. It could, for instance, be rewarding with thorough analyses on what effect different occurrence spans have on the discrimination ability for individual keywords.

The algorithm for the KEF tool was not developed for any specific domain. There is, however, a possibility that decisions made in regards to the algorithm, had been different if we had looked at

documents from other domains than the DigLib collection. For example, the whole noun phrases might had been considered as suitable index terms.

The de-hyphenator module in KEF preserves hyphens in compounded words (e.g., *Chow-Chow*). However, if this hyphen occurs at the end of a line, the hyphen will be removed. In phrases like *barn- och ungdomslitteratur*, we decided to remove the hyphens, which incorrectly results in *barn* and *ungdomslitteratur*. Another issue that might cause an inadequate index is homonyms, i.e., words with different meanings that are spelt in the same way (e.g., krona (Swedish coin) and krona (kings crown)). Such words are treated as having the same meaning. There seems to be no way around these problems unless making a program that understands the semantics of texts.

What other solutions are there for finding keywords? Can word length be used as a sole criterion (i.e., without using a tagger) for finding relevant words? We performed some preliminary studies on the issue. These indicate that there are possibilities, but at the same time the result varies from document to document. For example, one text will produce a potential list of keywords when the word length is set at minimum 17 letters, another text will not have any words that long. This could possibly be solved by comparing the length of each word to the mean word length of the document. It also seems that texts containing fewer long words are written in a more narrative style (Karlgren & Cutting, 1994), and it ought to be more difficult to find keywords in that kind of texts even manually.

A non-trivial advantage when using the Swecg tagger in the indexing algorithm, is the lemmatisation, i.e., different inflections of a noun or a verb are regarded as occurrences of the same word (for example, *råtta*, *råttorna* and *råttans* are in the final output from KEF treated as three instances of *råtta*).

About Dienst

The system Dienst was deployed for users wanting to retrieve known documents. By providing a search on bibliographic records, it looks for titles and authors quickly, without bothering with the full document texts. There is, however, one shortcoming with Dienst; it can not correctly interpret, for instance, the letters *å*, *ä*, and *ö*.

When indexing, Dienst disregards all strange characters. The letters *å*, *ä* and *ö* fall in this category. A word like *språkvårdare* will by Dienst be indexed as *spr*, *kv* and *rdare*, as three separate words. All indexed words consist of at least two letters, i.e., one-letter words are ignored by the system. When searching on *översättare* (a word including, for Dienst, strange characters) the search is performed like a Boolean AND search, where titles including *vers* AND *ttare* are retrieved. This search would retrieve the correct documents, and no others, as the database at the moment does not include any other words containing both *vers* and *ttare*. The problem does not, however, pass unnoticed in all cases. For example, a search on *förord* (interpreted as *rord* - the single letter f is disregarded, and *ö* is ignored) will also retrieve a document called *Dansk radiærordbog* (indexed as *radi* and *rordbog*) meaning that *rordbog* is a hit as it begins with *rord*. *Fårord* and *förord* will retrieve the same documents (the search in not case-sensitive). A search term *rtiden* will retrieve a document containing the word *nuförtiden*, while simply stating *tiden* will not get any matches.

As multi-linguality is an important aspect in DigLib, the above discussed issue is rather unfortunate. This problem is, on the other hand, probably common to most search engines, since only a few are developed in consideration of the Scandinavian languages. We assume that adjusting

the codes in Dienst, in order to correct this deficiency, would not be an insurmountable problem, but outside the scope of our task.

About our Thesis Work

By implementing a platform for DigLib, the foundation for further work within the project was laid. By studying how our content handlers and scripts in sicsDAIS work, a future administrator of the system should not have any major problems plugging in a new search or text analysis tool. This is done by finding out where data goes into and out from the tool, joining these to a content handler (leading the data into and out from the system, and to and from the external program), and adding a line in the script. In this way information seeking tools can be added either on a permanent basis or for testing and evaluating.

So far the DigLib document collection is quite small, which to some extent prevents meaningful text analyses to be performed. In addition the inferior quality of the texts, due to the OCR processing, further limits certain text analyses. There are today concrete plans to increase the collection, and developing methods for OCR corrections, within the SICS DigLib project. This will make it more useful as a test collection.

Tools

Most text analysis tools are developed for the English language. Although the Swedish and the English languages are relatively close to each other regarding syntax and morphology, they differ in several crucial issues (e.g., Swedish word compounding). It is essential that researchers in Sweden look more closely into the matter of developing tools for the Swedish language, as it is most unlikely that this issue will be given any attention by researchers elsewhere. Although many details in Information Retrieval tools are language specific, some issues are, however, common for all languages. Progress made in one language can most likely benefit others. Developing robust techniques for any kind of information retrieval is, however, not altogether easy to accomplish. van Rijsbergen (1979) concludes that there will not be any decisive progress of IR techniques, unless major effort is put in the linguistic areas such as solving the semantic aspect of, for example, automatic indexing and machine translation. Nowadays this standpoint is even more topical because of the extensive development in computer hardware, and a bias towards experimenting with new and revolutionary technology, rather than towards sophisticated IR algorithms.

The Platform

The choice of using the sicsDAIS agent interaction system for the implementation of the digital library, was beneficial for our thesis work, but must also be considered in a wider perspective. The research and development of various digital library techniques is flourishing world wide, and a large number of projects has the goal to incorporate the latest agent technology with that of digital libraries. For example, University of Michigan, has within the framework of the US Digital Libraries Initiative, worked on creating and reviewing agent-based architecture for digital libraries (Atkins et al., 1996). From this point of view, our digital library platform can be viewed as being well ahead, although for the time being in a small scale. If, in the future, wanting to expand DigLib, the implementation through sicsDAIS will probably turn out to be an advantageous architecture. DigLib will not merely be confined to functions deployed within SICS, but can easily plug-in agents providing other kinds of services.

DigLib

DigLib meets three of the four prototypical information seeking interactions proposed by Belkin and Carballo (1998): “finding a known information object”; “recognising useful information objects by scanning through an information resource”; and “determining the content/structure of a set or collection of information objects”. The fourth interaction “evaluating the usefulness of information objects” cannot be met by adding a single option to the left hand menu of the platform, since this rather concerns different aspects of individual documents or sets of documents. Evaluating a document in regard to a user’s information need, can for example be accomplished by comparing the document to other documents in the collection or by custom-made summarisations of retrieved document/s.

Although digital libraries lack problems that traditional libraries have, other problems constantly arise. The most important is perhaps the legal matters concerning intellectual properties and publisher rights (Griffin, 1997; Stefik & Lavendel, 1997). There are many advantages having digital documents easily distributed and accessible on networks, but there are also drawbacks: the risk of people ignoring copyrights cannot be neglected. Another problem common for the whole computer area is that of hardware failure and maintenance lulls - leading to longer or shorter standstills - preventing users to access the digital library.

The research in digital libraries spans over many different areas, ranging from hardware technology, through the subject of human machine interaction, to issues concerning linguistic features. The US Digital Libraries Initiative, described in section 2.2, certainly verifies this fact. Libraries have been pillars of the cultural community since humankind first started writing, and adopting the concept of a digital library should therefore be rather straightforward. Progress made in the field will most likely have an impact on society in general, as digital libraries more and more will become a natural source of information. A digital library can, however, not replace a traditional library as a physical place when it comes to its atmosphere, and the imbibing of knowledge.

References

Anderson, J.R. (1995). *Cognitive Psychology and its Implications.* (4th ed.). New York: W.H. Freeman and Company.

Atkins, D.E., Birmimgham, W.P., Durfee, E.H., Glover, E.J., Mullen, T., Rundensteiner, E.A., Soloway, E., Vidal, J.M., Wallace, R. & Wellman, M.P. (1996). *Toward Inquiry-Based Education Through Interacting Software Agents*. From Computer theme issue on the US Digital Library Initiative. May.
URL: http://www.computer.org/computer/dli/r50069/r50069.htm

Baldachi, M.B., Biagonini, S., Carlesi, C., Castelli, D. & Peters, C. (1998). Implementing a Common User Interface for a Digital Library: the ETRDL experience. Preliminary draft. Paper presentation at: *8th ERCIM/DELOS Workshop on User Interfaces for Digital Libraries*. Stockholm, October.

Beekman, G. (1999). *Computer Confluences.* (3rd ed.). Addison Wesley Longman, Inc.

Belkin, N.J. & Carballo, J.P. (1998). Understanding and Supporting Multiple Information Seeking Strategies, a TIPSTER Phase III Research Project.
URL: http://www.scils.rutgers.edu/tipster3/18month/index.htm

Berners-Lee, T.J., Calliau, R. & Groff, J.F. (1992). The World Wide Web. In: *Computer Networks and ISDN Systems.*

Birn, J. (1995). *A Syntax-Geared Approach to Determiners and Pronouns in Swedish Constraint Grammar*. Department of General Linguistics. University of Helsinki.

CGI. (1998). Common Gateway Interface. URL: http://hoohoo.ncsa.uiuc.edu/cgi-1.1/intro.html

Charlton, P., Chen, Y., Mamdani, E., Pitt, J., Espinoza, F., Olsson, O., Waern, A. & Somers, F. (1997). Open Agent Architecture supporting Multimedia Services on Public Information Kiosks. In: *Proceedings of Practical Applications of Intelligent Agents and Multi-Agent Systems*. London, April.

Cornell. (1998). Cornell University. URL: http://www.cornell.edu/

Croft, W.B. (1993). Knowledge-based and statistical approaches to text retrieval. *IEEE EXPERT - Intelligent Systems & their Applications.* 8(2): 8-12, April.

DARPA. (1998). The Department of Defense Advanced Research Projects Agency.
URL: http://www.darpa.mil/

Davis, J.R. & Lagoze, C. (1994). A protocol and server for a distributed digital technical report library. URL: http://cs-tr.cs.cornell.edu/Dienst/UI/2.0/Describe/ncstrl.cornell/TR94-1418?a

Dienst. (1995a). NCSTRL Documentation.
URL: http://www.ncstrl.org/Dienst/htdocs/Info/about-ncstrl.html

Dienst. (1995b). Here's what you need to Install and Administer a Dienst system.
URL: http://www.cs.ust.hk/Dienst/htdocs/admin_doc_menu.html

DLI. (1998a). The US Digital Libraries Initiative, Phase One. URL: http://www.dli2.nsf.gov/dlione/

DLI. (1998b). The US Digital Libraries Initiative, Phase Two. URL: http://www.dli2.nsf.gov/

DTD. (1998). How to read the HTML DTD.
URL: http://www.w3.org/TR/REC-html40/intro/sgmltut.html#h-3.3

Engcg. (1996). A Short Introduction to ENGCG. URL: http://www.lingsoft.fi/doc/engcg/intro/

ERCIM Workshop. (1998). *8th ERCIM/DELOS Workshop on User Interfaces for Digital Libraries*. Stockholm, October.

Espinoza, F. (1998). *sicsDAIS: Managing user interaction with multiple agents*. Licentiate thesis 98-007. Department of Computer and Systems Sciences. Stockholm.

Flanagan, D. (1997). *JAVA in a Nutshell* (2nd ed.). Sebastopol: O'Reilly & Associates.

Genesereth, M., Fikes, R. (1992). *Knowledge Interchange Format, Version 3.0 Reference Manual*. Technical Report Logic-92-1. Computer Science Department. Stanford University.

Griffin, S.M. (1997). Digital Libraries and the NSF/DARPA/NASA Digital Libraries Initiative. In: Raitt, D. (1997). *Libraries for the new millennium. Implications for managers.* (Ed.). London: Library association publishing.

Hansen, P. (1997). *An Exploratory study of IR Interaction for User Interface Design*. Master thesis 1997:81. Högskolan i Borås.

HTML. (1998). 4.0 Specification, W3C. URL: http://www.w3.org/TR/REC-html40/

Hulth, A. & Jonsson, A. (1998). One Interface - Several Information Seeking Tasks. Poster Presentation at: *8th ERCIM/DELOS Workshop on User Interfaces for Digital Libraries*. Stockholm, October.

Icesoft. (1998). URL: http://www.icesoft.no/Doc/ICEBrowserLite/api/index.html

Ingwersen, P. (1992). *Information Retrieval Interaction*. Taylor Graham Publishing.

JFC. (1998). JFC - Documentation. URL: http://java.sun.com/products/jfc/docs.html

Jones, S., Cunningham, S.J. & McNab, R. (1998). An Analysis of Usage of a Digital Library. In: Nikolaou, C. & Stephanidis, C. (1998).

Justeson, J. & Katz, S. (1995). Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural Language Engineering 1 (1) 9-27.* Cambridge University Press.

Karlgren, J. & Cutting, D. (1994). Recognizing Text Genres with Simple Metrics Using Discriminant Analysis. *Proceedings of COLING 94, Kyoto*. In: the Computation and Language E-Print Archive: cmp-lg/9410008.

Kohonen, T. (1997). *Self-Organizing Maps.* (2nd ed.). Springer-Verlag Berlin Heidelberg New York.

KQML. (1998). KQML as an Agent Communication Language. URL: http://www.cs.umbc.edu/kqml/papers/kqml-acl-html/root2.html

Källgren, G. (1984). *Automatisk exerpering av substantiv ur löpande text. Ett möjligt hjälpmedel vid datoriserad indexering?* IRI-rapport 1984:1. Institutet för Rättsinformatik. Stockholm University. (In Swedish.)

Källgren, G. (1992). *Making maximal use of surface criteria in large-scale parsing: the MorP parser*. Papers from the Institute of Linguistics. (Publication 60). University of Stockholm.

Lisp. (1996). Common Lisp Hyper Spec. URL: http://www.harlequin.com/education/books/HyperSpec/FrontMatter/Chapter-Index.html

LOC. (1998). Library of Congress. URL: http://www.loc.gov/

Luhn, H.P. (1959). Auto-Encoding of Documents for Information Retrieval Systems. In: Boaz, M. *Modern Trends in Documentation*. (Ed.). London: Pergamon Press. (Reprinted in: Schulz, C.K. (1968). *H.P. Luhn: Pioneer of Information Science, selected works*. (Ed.). New York: Sparta.).

Lynch, C. & Garcia-Molina, H. (1995). Interoperability, scaling, and the digital library research agenda. *Information Infrastructure Technology and Applications (IITA) Digital Libraries Workshop*. May. URL: http://www.ccic.gov/pubs/iita-dlw/

NASA. (1998). The National Aeronautics and Space Administration. URL: http://www.nasa.gov/NASA_homepage.html/

NEH. (1998). National Endowment for the Humanities. URL: http://www.neh.gov/

Nikolaou, C. & Stephanidis, C. (1998). *Proceedings of the Second European Conference on Research and Advanced Technology for Digital Libraries.* (Eds.). Second European Conference, ECDL'98. Crete, September.

NIST. (1998). National Institute of Standards and Technology. URL: http://www.nist.gov/

NLM. (1998). National Library of Medicine. URL: http://www.nlm.nih.gov/

NSF. (1998). National Science Foundation. URL: http://www.nsf.gov/

Nylander, S. (1998). Scanning for DigLib. URL: http://www.sics.se/diglib/publications/stny_98/rapport.html

Olsson, F., Gambäck, B. & Eriksson, M. (1998). Reusing Swedish Language Processing Resources in SVENSK. In: *Proceedings of Minimizing the Effort for the Language Resource Acquisition.* European Language Resources Association.

Paepcke, A., Cousins, S.B., Garcia-Molina, H., Hassan, S.W., Ketchpel, S.P., Röscheisen, M. & Winograd, T. (1996). *Using Distributed Objects for Digital Library Interoperability*. From Computer theme issue on the US Digital Library Initiative. May. URL: http://www.computer.org/computer/dli/r50061/r50061.htm

van Rijsbergen, C.J. (1979). *Information Retrieval.* (2nd ed.). URL: http://www.dcs.gla.ac.uk/Keith/Preface.html

RFC 1729. (1994). Z39.50. URL: http://www.landfield.com/rfcs/rfc1729.html

RFC 1807. (1995). URL: ftp://nic.merit.edu/documents/rfc/rfc1807.txt

RFC 2068. (1997). HTTP. URL: http://www.nada.kth.se/projects/prup98/web_proxy/rfc2068.html

RTF. (1997). RTF Version 1.5. Rich Text Format (RTF) Specification and Sample RTF Reader Program. Application note. Microsoft Corp.

Salton, G. (1989). *Automatic Text Processing.* Addison-Wesley Publishing Company.

Sampson, G. (1985). *Writing Systems.* Stanford: Stanford University Press.

Schatz, B. & Chen, H. (1996). *Building Large-Scale Libraries.* From Computer theme issue on the US Digital Library Initiative. May. URL: http://www.computer.org/computer/dli/

Schatz, B., Mischo, W.H., Cole, T.W., Hardin, J.B., Bishop, A.P. & Chen, H. (1996). *Federating Diverse Collections of Scientific Literature*. From Computer theme issue on the US Digital Library Initiative. May. URL: http://www.computer.org/computer/dli/r50028/r50028.htm

Shockwave (1999). Shockwave 7 and Flash Player. URL: http://www.macromedia.com/shockwave/productinfo/whyuse/

SITI. (1998). Svenska IT-Institutet. URL: http://www.siti.se/

SGML. (1998). Introduction to SGML. URL: http://www.w3.org/TR/REC-html40/intro/sgmltut.html#h-3.1

Smeaton, A.F., Morony, M., Quinn, G. & Scaife, R. (1998). Taiscéalaí: Information Retrieval from an Archive of Spoken Radio News. In: Nikolaou, C. & Stephanidis, C. (1998).

Smith, T.R. (1996). *A Digital Library for Geographically Referenced Materials*. From Computer theme issue on the US Digital Library Initiative. May. URL: http://www.computer.org/computer/dli/r50054/r50054.htm

Stefik, M. & Lavendel, G. (1997). Libraries and Digital Property Rights. In: Peters, C. & Thanos, C. *Proceedings in Research and Advanced Technology for Digital Libraries*. (Eds.). First European Conference, ECDL'97. Pisa, September.

Sun Microsystems, Inc. (1998). URL: http://java.sun.com/products/OV_jdkProduct.html

TIFF. (1992). Revision 6.0. Aldus Developers Desk. Aldus Corporation. URL: http://www.ndg.com/tf1.htm

Wactlar, H.D., Kanade, T., Smith, M.A. & Stevens, S.M. (1996). *Intelligent Access to Digital Video: Informedia Project*. From Computer theme issue on the US Digital Library Initiative. May. URL: http://www.computer.org/computer/dli/r50046/r50046.htm

Wilensky, R. (1996). *Toward Work-Centered Digital Information Services*. From Computer theme issue on the US Digital Library Initiative. May. URL: http://www.computer.org/computer/dli/r50037/r50037.htm

Xerox. (1998). Xerox Palo Alto Research Center. URL: http://www.parc.xerox.com/parc-go.html

Zetie, C. (1995). *Practical User Interface Design: Making GUIs Work*. London: McGraw-Hill.

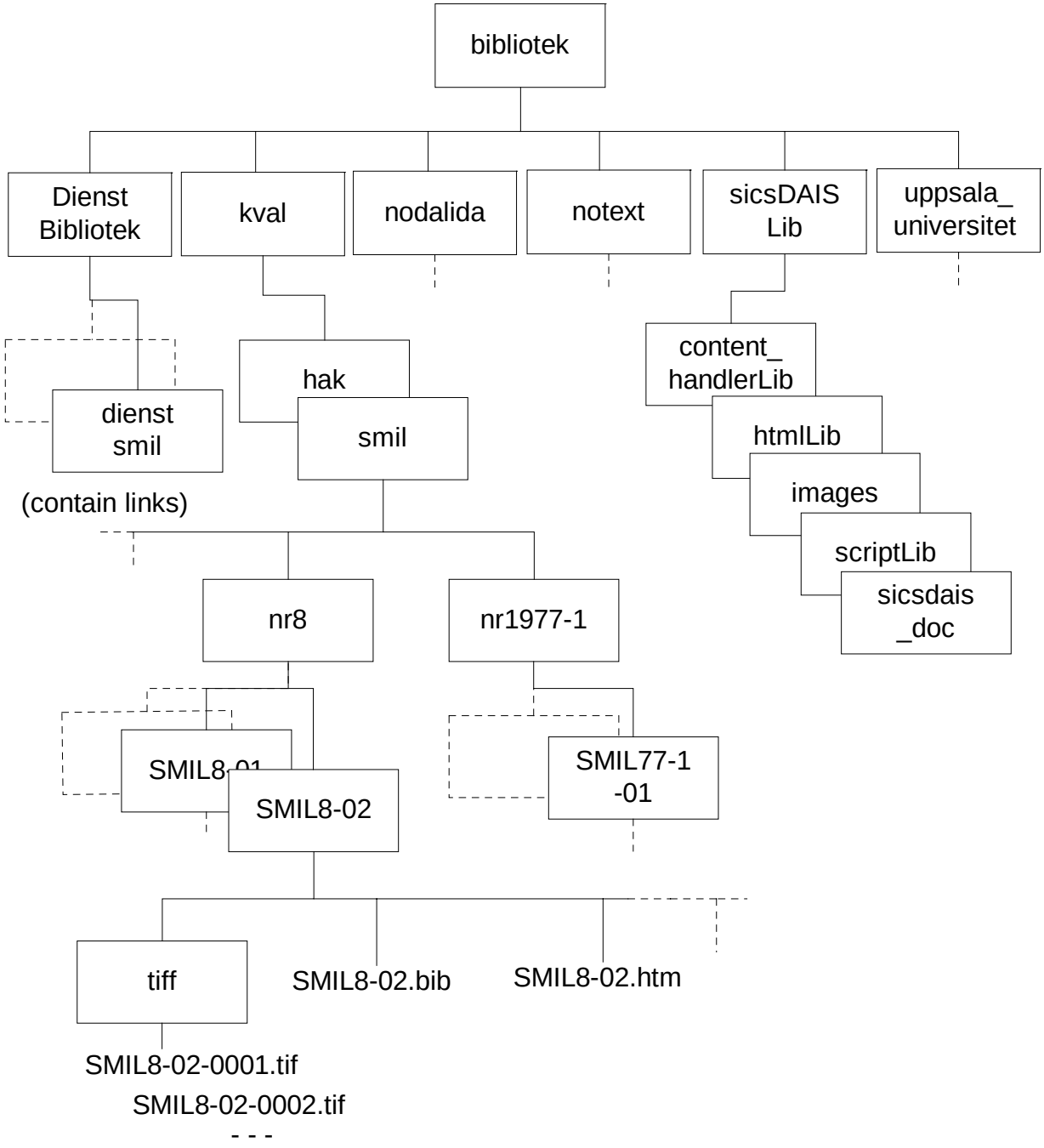
Appendix A

The research groups in the Digital Libraries Initiative Phase One (DLI, 1998a) are

- *University of California, Berkeley*. This project studies technology for intelligent access to massive, distributed collections. This requires a broad technical plan that deals with document image analysis; natural language analysis and computer analysis for useful information extraction; user interfaces and tools for enhanced access to multimedia information; and improved protocols for client program interaction with organised collections. (Wilensky, 1996.)
- *University of California, Santa Barbara*. The Alexandria project develops a digital library that provides access to large and diverse collections of maps, images and other spatially indexed information. The project aims to enable the presentation of multiple interfaces to serve users with different backgrounds and needs. For example, the needs and expectations of schoolchildren, when interacting with an interface, will not be the same as for scientists. (Smith, 1996.)
- *Carnegie-Mellon University*. The digital video project at Carnegie-Mellon University explores how multimedia digital libraries can be established and used. It applies multiple techniques for content-based searching and video-sequence retrieval. To enable successful segmenting and indexing, and search in diverse video collections, there is a need of combined interaction regarding image, speech, and natural language understanding technology. (Wactlar, Kanade, Smith & Stevens, 1996.)
- *University of Illinois at Urbana-Champaign*. This project has its focus on information infrastructure and on semantic search of technical documents on the World Wide Web. One issue is making distributed collections of diverse materials appear to be a single integrated collection. The project is also developing usable Web technology in order to improve Web search beyond full-text retrieval. (Schatz et al., 1996.)
- *University of Michigan*. This project is concerned with the creation and evaluation of an agent-based architecture for digital libraries. I.e. to co-ordinate the operations of agents that provide library services, automated team formation, information search-space structuring, and market-based resource allocation. (Atkins et al., 1996.)
- *Stanford University*. The digital library project at Stanford studies interoperability and approaches through distributed object technologies to enable uniform, easy access to dispersed network sources and collections. They have found that a distributed object framework gives clients and servers the flexibility to manage their communication and processing resources effectively. (Paepcke et al., 1996.)

Appendix B

Below, the defined structure of the collection is shown. The DigLib collection consists of the directories: *kval*; *nodalida*; *notext*; and *uppsala_universitet*. The Dienst directory, *DienstBibliotek*, contains the links for Dienst. The sicsDAIS directory, *sicsDAISLib*, contains among other things, the content handlers and the scripts created for DigLib. The directory containing the Swedish documents' URLs for KEF (*linkSwe*) is not shown in this picture, but its place is between *kval* and *nodalida*.



Appendix C

KEF removes all words that consist of three letters or less. The removed words are listed below (thus the words are those that occur in a Swedish document in the current collection and are marked as *noun* by Swecg).

act	don	hår	min	rel	ucp
adv	dot	ia	mit	rep	uds
akt	dås	iag	mo	rng	udu
al	död	id	mod	ro	unc
ala	ed	ide	moo	rot	uop
alg	edi	idé	mor	rum	ur
alt	ego	ie	mun	rz	usa
ama	egs	iet	mvs	rå	use
and	ei	iga	mål	råd	utg
any	en	igh	mån	rö	vad
ar	ene	il	na	sak	vak
arb	eng	ipp	nar	see	val
are	eq	ir	nav	sej	var
arm	est	iri	neq	sem	ve
art	fa	is	new	set	vlc
as	fan	isä	nfi	sg	vnr
asp	far	its	nil	sh	voi
ass	fas	ja	nit	she	vol
ave	fe	jcl	nod	sit	vti
avf	fel	jf	not	sjö	väg
bal	few	job	now	sko	vän
bar	fik	knä	ntv	slå	wa
bas	fil	ko	nät	sne	war
bet	foc	kod	obj	so	was
bi	fog	kol	obs	spe	way
bil	fot	kor	och	szh	we
bit	fru	kun	ocn	säl	wet
bo	fx	kö	ode	sär	who
bok	får	kön	off	sök	why
bor	för	kör	og	tab	wi
bso	gar	lag	ohc	tag	wo
but	gem	led	omr	tak	won
by	gen	lem	one	tal	wäl
bär	get	lex	or	tax	xh
böj	gik	lfg	ord	ten	xi
can	gil	lis	ort	th	yet
car	gir	lit	os	tid	yh
cdr	giv	liv	osf	tij	yl
ch	go	ll	our	til	you
cit	gom	llc	oxe	tin	yta
clt	got	lo	par	tit	z^
cmm	gud	lob	pay	tlf	za
col	gut	log	pi	tof	zo
con	gwb	lot	pl	tok	ära
cr	had	lov	pn	tom	äx
cs	hak	lun	pnf	ton	åhr
dag	hav	lån	pom	too	åk
dal	hen	lås	pos	top	år
dan	her	lön	pph	tor	ås
dcg	him	maj	prm	tro	éik
dcl	his	man	psi	tsf	öf
def	ho	mat	rad	tte	öga
del	hom	max	ram	tur	öl
dfu	hop	may	ras	tva	ör
di	hov	mi	re	two	öre
did	hus	mil	ref	typ	
dlt	här	mim	reg	tåg	

Appendix D

In the left hand column below, the content handlers created for sicsDAIS to handle the DigLib platform are listed. In the right hand column, the methods for each content handler are listed together with its parameters. (For a more detailed description of the different content handlers, see section 4.2.) The methods listed are sicsDAIS level events which can be stated in the script (not the internal methods executed only within the content handler class).

Content Handlers	**Methods & Parameters**
• buttonLabel	(setButton ”yellowLarge” | ”yellowSmall” | ”redLarge”) (changeFrame ”String url”) (setLabel ”String s”)
• CHPicture	(setMotive ”verticalLine” | ”horizontalLine” | ”logo” | ”welcome” | ”quit”) (pressButton) [for ”quit”]
• dienstCH	---
• emptyCH	---
• helpCH	(setText ”String s”) (turn)
• HTMLViewerCH	(showText ”String s”) (showBrowser) (hideBrowser) (otherContextEval)[calls the wrapped IceBrowser object]
• infoBox	(setInfoText ”String s”)
• kefCH	---
• webCH	(webPage ”String url”)

Appendix E

Listed below are the scripts created for sicsDAIS to handle the different events trigged by clicking the left hand menu buttons of DigLib. The different content handlers responsible for this are stated in parenthesis after each script.

Scripts	This script handles
• aboutScript	about (creates and lays out infoBox and CHPicture)
• browseScript	browse the collection (creates and lays out helpCH, emptyCH, CHPicture and HTMLViewerCH through webCH)
• dienstScript	retrieve known document (creates and lays out CHPicture, dienstCH and helpCH)
• infoScript	help desk (creates and lays out infoBox and CHPicture)
• kefScript	index words (creates and lays out kefCH, CHPicture and helpCH)
• newScript	new documents (creates and lays out CHPicture and HTMLViewerCH through webCH)
• mainDigLibScript	initiation script - left menu (buttonLabel, emptyCH, HTMLViewerCH and CHPicture)