The Paradigma Web Harvesting Environment

Ketil Albertsen

The *Paradigma* project at the National Library of Norway

Abstract. The paper describes a proposed software design covering all computerized operations related to the acquiring, cataloging and long time archival of legal deposit documents in digital form. The design is nearing completion, and will be presented to the Paradigma project steering committee for evaluation later this year.

1 Introduction

The National Library of Norway has participated in several international projects addressing issues related to legal deposit of digital documents, e.g. Nedlib[1], Biblink[2], and Nordic Web Archive[3]. The present work is founded on experience from these projects.

The Paradigma project[4] was initiated in August 2001 to establish a complete production line for harvesting and archiving digital documents covered by the Norwegian legal deposit act. Since 1990, the law has mandated deposit of digital documents. However, tools and infrastructures have not been oriented towards Internet documents, so, deposit has mostly been restricted to digital publications physical media, such as CD-ROMs, and a few e-publications harvested manually.

This paper proposes a complete software infrastructure for digital legal deposit handling, and addresses a broad spectrum of issues that may be raised in a production system, such as access control, version and variant management, user access tools etc.

2 Relationship to similar projects

Some national libraries have harvested web documents for legal deposit for as long as five years. In Norway, the first trial harvesting was carried out in December 2002, the second one in July 2003. Web archiving projects, and tools developed for this purpose, generally falls in one of two groups: Projects like the Danish Indoreg[5] and the Greenstone[6] toolkit developed by University of Waikatop, New Zealand, has chosen a traditional approach for collection establishment, using qualified personnel to select, organize and catalogue a fairly small fraction of the available material. Web harvesting done in projects such as the Internet Archive[7] and the Swedish Kulturarw3[8] has focused on completeness rather than qualified selection, but with little emphasis on collection handling and user access facilities.

Paradigma's goal is to build a total system capable of complete harvesting which is designed for partially automated archive management through programmed functions for identifying and associating separate parts making up a complete document or documents available in multiple versions or variants. Tasks which traditionally were labor-intensive manual tasks will be semi automated: A rule-based automatic procedure will extract whatever metadata is available from the document itself and other sources, perform a preliminary selection and ranking, and present a ranked list of candidates to a librarian who will do the finishing work.

The architecture also encompasses user access tools and functions for interoperation with an external library catalog system.

While most of the issues discussed in this paper have been addressed by various web-harvesting projects, we are not aware of any other project with a comprehensive, total approach to these problems at a comparable level with Paradigma.

3 Information Gathering

Paradigma's primary goal is to collect all digital documents covered by Norway's legal deposit act. Information is gathered from multiple sources: Internet harvesting using HTTP, FTP and other protocols, receiving email publications and NetNews discussion groups, and obtaining "dumps" of complete databases that have been made generally available across the Internet.

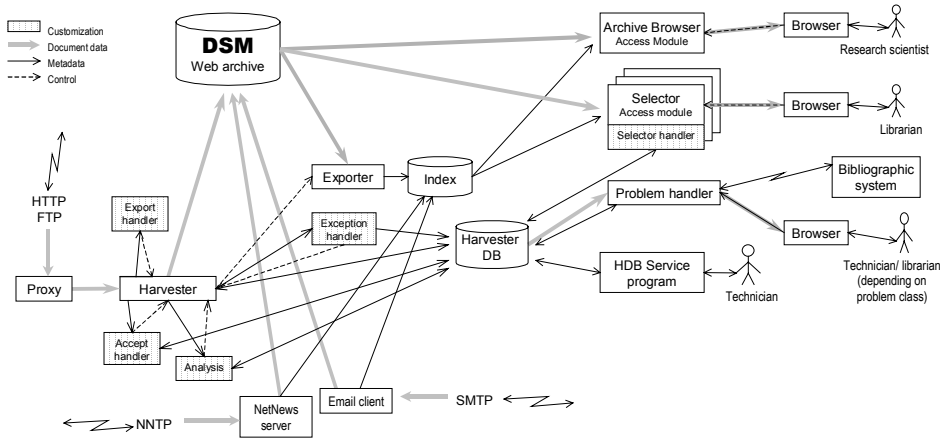

Fig. 1. Proposed software overview

Long-term storage is handled by the Digital Storage Vault (*DSM)* of the National Library of Norway, currently storing 22 TByte on disk and 100 TByte on tape. Digital objects may be added to or read from the archives; object modifications are strictly controlled – essentially, DSM is a read-only archive.

DSM handles *Digital Objects* (DOs). A DO is a uniquely identified and logically homogenous object. If it has multiple components, these are, presumably, alternate representations of the same logical information. Metadata about the DO is stored with the object in DSM.

Network documents are loaded into DSM after complete processing, with the possible exception of bibliographic cataloguing. A customer searching the archive will access the object from DSM. The text contents of DOs are full indexed; this index is not considered to be a part of DSM, but is used as an access path to DSM objects. A bibliographic catalog may also serve as an access path to DSM objects.

3.1 Harvester Database

The harvester database is not part of the DSM; it holds working data for the gathering process. Parts of the information in the base are saved as metadata in the DSM when an object is archived. The database scheme maintains all information abut all relevant Internet servers, directories, URLs (with query part and cookie information), logical document structures, recognized data types, language codes etc. The database consists of roughly 50 SQL tables of greatly varying size.

3.2 Web Harvesting

Crawling. A crawler starts with a few initial URLs. The addressed object is retrieved, parsed and searched for links to follow. In our environment, each retrieved object is evaluated for archival and further search. Objects not covered by the legal deposit act are discarded. Links in selected documents are processed like the URLs in the initial set. This procedure is applied recursively.

Links embedded in e.g. Macromedia Flash movies or created by a script may go undetected from HTML parsing. In the future, we may learn to simulate all possible script execution paths and all possible user input classes automatically; this is not possible today. Objects with potential problem links, e.g. Javascript window.open() calls with a variable first (URL) argument are reported as exceptions. The operator will review these web pages routed through an HTTP proxy, whose only function in this context is to register all URLs requested. Although valid URLs may still go undetected, we expect the majority of problem URLs to be detected this way.

Selection. When selecting objects for archival and further search, the *entire web page* (a *document)* is evaluated as a whole, not as individual web objects (files): All images, sounds, text etc. are collected before a decision is made.

Documents published in **.no** or domains owned by Norwegian institutions/ individuals are selected; ownership information is obtained through the **whois** service. Other documents are selected if the language is Norwegian or Sami, based on HTTP/HTML language tags, a significant frequency of national letters (æ, ø, å, Sami accented characters), code sets tailored to Norwegian/Sami, or a large fraction of the words are present in Norwegian/Sami dictionaries.

Result analysis after a trial harvesting round, collecting 3.1 mill documents from **.no**, indicates that web page authors frequently use misleading or unstandard language

and character set tags. Unknown tag values reported as exceptions, and the operator may decide future accept/reject of the tag as indicating Norwegian/Sami language. Advanced language recognition may be added later.

A document may *partially* qualify for selection: Some components qualify, others don't. A "goodness" value is calculated by a weighing function, considering factors such as information type, domain, relative size etc. of each component. If the value exceeds a specified threshold, the entire document is selected. Components which are not by themselves selectable may be marked for less frequent (or no) refresh. The weighing function can be customized. Typical customization would be to change the default handling of Sami language documents found in the **.se**, **.fi** and **.ru** domains.

Our *frontier* (the set of evaluated but rejected documents) has depth 1: Links in rejected objects are ignored. The design allows a deeper frontier: The frontier document itself is rejected, but links are evaluated *n* steps further.

Normalization. HTML permits different character representations; e.g. 'Ø', '&#216' and '&#xD8' are equivalent. Non-ASCII character sets add further alternatives. Line terminators vary: 'LF', 'CR/LF' and others. Files may be compressed and encrypted. Before any text is evaluated (e.g. searched for links), a normalized representation of the text contents is created, where format and representation differences are removed.

If any significant text fraction is generated by scripts, the script must be executed to create a normalized form. Scripts depending on user input are exceptions that require manual operator assistance. We hope to be able to add automated script evaluation at a future time, avoiding manual handling for simpler cases.

Although a normalized text representation is generated, the original form is archived as the primary copy. The normalized form is a "work copy" only.

3.3 Possible Problems With Web Pages

Client side scripts. If no exceptional situations are detected, script code is retrieved and stored as part of the document. Exceptions include user input dependent code, script generated links whose value cannot be determined automatically etc. Exceptions are reported to an operator for manual handling of the document.

HTTP query part. A URL trailed by a question mark and a sequence of field/value pairs *may* indicate that the web page is generated on the fly from a database, and is reported as an exception. The operator may decide to treat the query part as an URL extension for future refreshes, or the query part may be modified or removed.

If the operator finds reasons to believe that the page is generated from a database, he may suspend all access either to this URL, to URLs having a common path as this one, or to the entire web site, and take the appropriate actions to have the information deposited as a complete database, rather than as individual web pages.

HTML *FRAME*s. In principle, each frame displays independent documents. In practice, documents frequently interact closely through script code. So, for selection and archiving, the collected contents of all frames are treated as one document.

Shared components. A component, often a style sheet, image or background sound, may be shared by multiple documents. The component is retrieved and archived only

once, but it is included in the definition of all relevant documents. If an update of the component is detected, all relevant documents are refreshed to ensure internal document consistency, even if the refresh time for the document is not yet reached.

Cookies. The first access to a URL is made with *no* cookies, even if the URL is within the scope of known cookies. If a cookie is returned, the access is retried with this cookie value, and returned data is compared. If equal, and the second access returns the *same* cookie value, cookies are ignored for later refreshes. If different data is returned, this is reported as an exception. The operator may decide to ignore the cookie, or to define the cookie as a URL extension: On refresh, the URL is accessed several times, with each known cookie values, to obtain multiple variants of the page.

Streaming media and incrementally transferred data. Streaming media (e.g. sound or video) and data types transferred piece by piece to the browser (e.g. geographical maps) are frequently managed by plugins, and no functions for storing the information are provided. The web site may keep a dialog with the plugin, demanding information, e.g. for authentication, not externally available. This may prevent any access to the data not provided by the plugin, and thwart automated legal deposit.

If data cannot be retrieved as a closed unit, suitable for playback without web site communication, the URL is reported as an exception. The operator may then suspend the URL for future updates and request the data deposited as a database.

Unknown MIME types. More than 400 distinct MIME types have been encountered in web pages. New, unknown types are reported as exceptions. The operator should ascertain that a plugin or format specification is available for data interpretation.

If no format specification is available, and interpretation is only possible through a plugin in binary format, long-term access to the information can be guaranteed only if a full environment for the plugin can be emulated in the future. Although shown to be possible, at least in theory, this *cannot* be guaranteed to be provided for all types.

3.4 FTP files, NetNews and email contributions

This design handles FTP files linked from web pages like objects obtained through HTTP. Metadata is usually sparse; FTP provides no "header lines" like HTTP; selection is based on host name, and, for text formats, language recognition.

Some institutions provide large FTP collections on servers typically named **ftp.**<*institution*>**.no**. Usually, most files have foreign origin, but they are collected, organized and made available to the Norwegian public from a Norwegian site. We suggest to archive such collections in full, but to keep refresh frequency low.

All NetNews groups generally available in the **no.*** hierarchy are considered covered by the Norwegian legal deposit law, but no groups outside of **no.***. Although Norwegian posters are very active in international groups, automatic identification of parts of the contents of international groups relating to Norway is not realistic today.

"Mailing lists", used by a number of net periodicals and for discussion fora with a more restricted distribution than NetNews, always require an explicit request/ subscription. Searching for generally available lists resemble the search for new printed periodicals, and is mostly a manual operation. List owners may argue that their list is not generally available; this is a familiar issue from the world of printed

periodicals and should be handled similarly. Mailing lists, which the publisher accepts to distribute to the National Library, should be archived like other digital publications.

3.5 Restricted Access Resources

A right holder may restrict document access. If the restriction level is high, the document is not considered generally available, and legal deposit is not applicable. Such documents are not archived by the National Library. Moderate restrictions levels, such as an access fee, do not exempt a document from legal deposit. All access restrictions are registered as metadata in DSM and taken into account by the access control system. One mechanism is used for all kinds of restrictions, regardless of restriction level, document source, and communication protocol.

Pay services. Documents obtained from a pay services should be kept in collections that requires a ticket issued by, or by permission of, the rights holder for access. ("Collection" and "ticket" is discussed in the chapter about access control.)

X-No-Archive. NetNews and email contributions may be marked with a request not to archive the entry permanently, even though the document is freely available for some time after publishing. Formally, X-No-Archive is a non-standard request that we are not obliged to honor, but its use has been widespread for years, and the Internet community more or less expects it to be treated as a standard header. X-No-Archive documents should be put in collections requiring special clearance (ticket) for access.

Cancelled entries. NetNews and email contributions may be explicitly cancelled, causing immediate invalidation. Canceling may also be used to replace an incorrect or incomplete contribution, or as a mechanism for stopping spam or virus from spreading. Cancelled documents should be put in collections requiring a special clearance (ticket) for access.

robots.txt. A robots.txt file specifies web pages not to be archived or indexed. This is used both for rights protection and to indicate that archival/ indexing is meaningless (e.g. a search result list). The harvester should put any harvested page covered by robots.txt in a collection requiring special clearance (ticket) for access.

3.6 Duplicate detection

Paradigma recognizes advantages of archiving a given object once only under a single identifier. The architecture is prepared for procedures to detect duplicates at all levels:

Web sites may have alternate names, either name used in URLs. A pure textual comparison would treat otherwise identical URLs using synonym host names as different. So, new host names are looked up in DNS, and the IP address/TCP port number used as the unique host ID, to prevent multiple accesses. Note, however, that the web server may use the host name part of the URL may to select which web pages to provide; this is termed *virtual hosting*, and the duplicate detection should be aware of this.

A host may have multiple IP addresses. If the harvester detects that two addresses return exactly the same set of documents, **whois** information for each address is retrieved and reported as an exception. The operator may confirm or reject the two addresses as being the same host. In some cases, one host serves as a "mirror" for another one, offering the same documents, yet being a distinct host. The operator may choose to disable all future accesses from mirror hosts.

If supplied by the server, MD5 checksum and Entity Tag values are saved when an object is harvested. If not supplied, the MD5 may be calculated locally. If the MD5 or Entity Tag matches a previously harvested object, the object is considered to be a second copy not to be archived again. The source URL and timestamp are then added to the already archived object.

MD5 checksums are also calculated for the normalized text representation (see above), filtering out representational differences. Equal values are handled in search results, presenting them as one logical document available in alternate formats.

Most web servers map URLs more or less directly to a file system path. If the path contains a ".." (up one level), the harvester will try whether a "coalesced" path returns the same object. If it does, this path is registered as a synonym to the original one.

Information about request redirection is stored in the harvester database, so that several redirections to the same web page do not cause multiple later refreshes.

MD5 based duplicate detection can be applied to FTP files. NetNews duplicates can be detected by the Message-ID (cross posted entries) or by MD5 checksum (multiposted entries).

3.7 Refresh Interval

Most large-scale harvesting projects make low-frequency (a few times a year) total sweeps, collecting *all* web pages available, whether or not an identical copy is already archived. Short-lived network objects may never be detected, and many versions of rapidly changing objects may be lost when this strategy is applied.

We suggest a continuous, low-intensity, "incremental" update. Each URL (with possible synonyms) has a last update time and a refresh interval, set when the URL is first encountered. The operator may override the system default (configurable, typically 1-3 months) for any web site, or parts of a web site. If a retrieved object indicates a long time has expired since the last modification, and/or it's expire time is far into the future, the interval is increased. If several refreshes cycles have all returned recently modified or a soon-to-expire object, the interval is reduced.

A librarian cataloguing a network document may explicitly set the refresh interval. A document's type, if known, affects the default interval – e.g. a research report needs less frequent updates than a periodical.

Using a low-intensity, incremental update strategy, the choice of "breadth first" or "depth first" search is less relevant: The time, rather than the location, is the steering factor. The incremental search will, however, consider *documents* as a whole, rather than individual URLs: When an URL is scheduled for refresh, all objects making up the document to which the URL belongs is refreshed as well to ensure internal consistency among the document parts.

3.8 Web Site Overload Prevention and Load Reduction

Total sweeps can easily overload a web site; *all* its web pages are usually retrieved. An incremental refresh strategy reduces this problem significantly, yet there may be hundreds or thousands of pending requests for large web sites.

To avoid site overload, we retrieve a small number of documents from a site, after which the site is put in quarantine for some time, blocking further requests. Both document count and quarantine duration are configurable, usually in the range of tens of documents and duration in minutes. Often, the delay before the next access is longer than the quarantine – the site is put last in the pending queue, so the harvester will often be busy with other sites.

Many objects are unmodified between refreshes. If the web site returns an MD5 checksum, Entity Tag or time stamp for the document, these parameters are added as conditions at the next refresh, to enable the site to return a simple "304 Not modified" status, instead of transmitting the possibly large object, when appropriate.

An increasing fraction of web pages are generated dynamically, making such optimizations irrelevant. In the long run we expect a significant fraction of databases, available through web sites, to be deposited as databases, so these pages will not be subject to harvesting from the web.

Duplicate detection may reduce the number of refresh requests that could cause multiple accesses to the same object. Particularly significant is the detection of synonyms for web sites, as this may affect all web pages on the site.

3.9 Exception and Error Handling

Exceptions cover a number of conditions where documents cannot safely be handled completely automatically: Unknown language/charset tag values, unknown MIME types, streaming data, script code generating variable links, FORMs etc.

Exception and errors are reported by inserting entries into tables in the harvester database. A HTML based interface presents the contents of the tables to an operator, and provides tailored handling facilities for each exception/error group. Different people will handle different groups – some problems are of technical nature requiring an engineer, librarians should handle other problems, some may be legal questions. We expect inspection of exception and error reports to be an everyday routine activity at the National Library.

"Exception" indicates that an object was retrieved, "error" that it was *not* retrieved. An error occurring while retrieving a component of a composite document is an exception for the document as a whole. When an exception or error occurs, further processing of the document is suspended until an operator has investigated the problem and decided further action. This may involve e.g. suspending further accesses to this site or URL, external actions e.g. to retrieve a database dump, supplying missing data such as a password etc. Frequently, the primary purpose of the operator's actions is to supply the harvester with information enabling it to handle the situation without operator intervention in future refreshes.

3.10 Customization

The Norwegian part of the web is estimated to hold roughly 10 million objects, and only 20% of these may remain unmodified through a year. Resource limitations permit manual handling of only a tiny fraction of these, and it is important to make the automated processing as good as possible. This requires a customization facility for the process that does not require the use of software development personnel.

Programming, from scratch, a custom function in a scripting language requires a certain competence level, although far less than a complete software developer education. *Modifying* selected parts of a script, e.g. the weighing factors of a ranking function, can usually be done by non-technical personnel after a brief introduction.

A custom function is usually written in a script language, allowing script modification on the fly. A generic interface to a script interpreter is defined: Each new script language requires the coding of an interface routine for interpreter activation and parameter transfer. This is a one-time operation. Any interpreter that can be activated from c code may be used; interfaces to JavaScript, PHP, Python and Perl interpreters are planned. If performance is essential, the function could be written in a compiled language such as c or c++. This would require relinking each time the function is modified, and programming in such languages usually requires a higher level of programming competence.

At well defined points in the processing of a document, the software checks for the presence of a customization function. If present, the function is called as a filter: The input and output format is identical; absence of a custom function is a null operation.

The input/output format is termed a *document vector* – an array of a number of properties of the document, such as percentage of Norwegian words, document size, number of pictures, amount of change since the previous version etc. For most customization, the vector gives sufficient information; reading the document text is not required (although it *can* be accessed). Typically, the custom function is called after an analysis, but before a decision is made, and the function may modify the vector values. The function may access the harvester database, e.g. for checking data about previous document versions, or access other bases (e.g. bibliographic ones). It may read, say, a list of publishers' web sites, and if the document URL addresses one of these, the "reliability" element in the document vector may be increased.

The vector follows the document as metadata through the system. For selection, the custom function may affect the decision by modifying the vector only. It may also return a final decision, overriding the default weighing of vector elements.

Similar customization is also available for functions making selections from archived documents, either for bibliographical work, or for customer searching in the library. In most cases, the purpose of the function is not to make absolute selection decisions but to modify the ranking value in a result list. A large part of the vector elements are oriented towards such use, reflecting properties that are not relevant in the initial selection at harvesting time.

3.11 Document Extent Definition

URLs locate document *components*, not documents. A librarian or archive user relates to complete documents – collections of components making up web pages, or larger units. If no hints about the extent of a document is available, the default, automated, document definition is the set of all graphical elements (text, images etc) required to make up one web page, background sounds, referenced style sheets and external script files, and other pages referenced through links with a `REL=` attribute clearly indicating that they belong to the same document, such as `contents` or `section`. This collection may be suggested as the unit for bibliographic registration and for access control.

A publisher or author requesting a unique ID for a network document, analogous to an ISBN, may specify another document extent. E.g. several web pages, each displaying a chapter of a report, may collectively be defined as one document, or components such as background sounds may be left out. A librarian entering catalog data for the document may also recognize several web pages (or other network objects) as one unit for bibliographical processing, and leave some elements out.

4 Document Archival

Unique identification and unique document contents. For storage purposes, each object needs a unique ID, serving as a primary key. For search, retrieval and reference, a document may need an externally visible ID. We distinguish between internal and external IDs; the latter identifies logical documents, which are compositions of objects, rather than individual physical objects[9].

A document may be implemented as an archive defined object referencing all the document's physical components. The external document ID is attached to this object, rather than to one specific component such as the top level HTML file. A similar mechanism represents a "Dynamic Document", which has a history stored as a sequence of references to snapshots ("Specific Documents"), and for documents in multiple variants (e.g. formats), which are unordered sets of document object references. Abstractions like FRBR[10] Expression and Work concepts are represented as archive defined "agent" objects.

An archive defined object that is created as a placeholder for an external ID *references* the components; it does not contain them. So, document definitions may overlap fully or partially, and one document may be a subset of another. In a deposit library, the document components are read-only and persistent, so the majority of consistency concerns often associated with overlapping definitions, are not relevant in our context.

Content extraction. Legal deposit documents are archived for long time storage; they should be accessible fifty years, one hundred years or more, from now. We cannot expect proprietary software to be available at that time. Content conversion *will* cause loss of information, so we suggest that the original bit stream, as received from the Internet, is always stored unmodified and considered authoritative.

For web objects in formats that we are not certain will be interpretable in the long term, secondary extract copies can be made and stored as alternate components in the Digital Object. For text in **.pdf**, **.doc** and other word processor formats, we create a pure text extract. Sound, graphics, photos and video in vendor-specific formats are converted to formats fully described in a recognized and publicly available standard specifications, that can be accessed in the future with a minimum of effort. We are fully aware that converted data should *not* be considered a valid replacement for the original, but only as a last resort the day that the original is no longer accessible.

For content extraction, we may require tools that are not available – the format owner may be unwilling to release format specifications or make such tools. In a few cases, independent developers have developed such tools by reverse engineering. If no extraction tools are made available, the affected objects may be lost for the future. We should note, however, that extracts could be made any time after the object was archived; the tool need not be available today to save the information for the future.

Encrypted and packed files. Decrypting a file may be treated similarly to content extraction: The encrypted file is treated as the primary copy, the decrypted one as a "last resort". The decrypted file may require special clearance for access; details depend on how the decryption key was obtained, and are handled manually by an operator.

Unpacking a single file may be treated like content extraction; both the packed and the unpacked file are stored. If a packed file unpacks to multiple files, they are stored as individual digital objects, individually accessible for cataloguing, indexing and user access. A document object may be created for the entire collection of files.

Virus handling Before an object is stored, it is scanned. If a virus is detected, access to the original object requires special clearance. If the virus can be removed, the cleaned version is saved as a document extract, as the one normally accessed.

Objects are virus checked at archival time, and rechecked 3-6 months later to detect new viruses that could not be detected at archival time. We suggest that after the second check, no further virus check is done. All objects, regardless of access protocol (HTTP, FTP, NNTP, SMTP...), are checked for virus.

Text indexing. Before being entered into the DSM, all objects are indexed. Text objects are full text indexed, objects of all data types are indexed by various metadata – bibliographical, data format parameters, retrieval parameters etc.

The retrieval key supplied to the index machine is the document identifier, asserting that the user will always see a search hit in a proper context.

Metadata handling. We estimate that due to resource limitations, at most 1% of all web objects will ever be evaluated by a librarian for cataloguing. So, the majority of objects may have no other metadata than those automatically extracted from the document and the communication protocol.

Bibliographic data, when present, is normally handled by an external system. To secure these data for the future, we make copies, format then in XML, and store them with the document. This copy is not used in normal operation, but serves as a long-term security backup against format changes etc.

5 Bibliographic Registration

With at most 1% of all network documents entered into the bibliographic system, the task of the automatic procedures is to select *which* 1%.

We term harvesting time decisions *Phase 1* selection: Norwegian/Sami documents are selected, others rejected. Phase 1 is fully automated. During this process, a document vector (see above) is created as a basis for the selection decision.

If a document is selected for archival in Phase 1, the document vector is augmented in a more thorough document analysis: The vocabulary size is determined, use of foreign or professional words is noted etc. Technical aspects are further analyzed - e.g., a ".../~name/..." part in a URL raises the probability that this is a personal home page. The analysis is done soon after harvesting, and is complete when the document is entered into the DSM. Up to this point, document handling is fully automatic.

The thorough analysis is the first part of the *Phase 2* selection. The second part is an automatic document ranking, calculated from the document vectors, when a librarian initiates a cataloguing tool. Different weighing factors may be applied for different tasks: A librarian who doesn't master Sami want Sami language documents to be ranked very low; a librarian handling children's books wants texts with a simple vocabulary to be ranked higher. Web documents from domains owned by recognized publishing houses may be ranked higher than those from other domains, etc.

The user interface displays the documents one by one, in ranked order, along with a form for entering bibliographic information. A librarian does the *Phase 3* selection, selecting which documents to register and which to skip over. Ideally, documents ranked first in the list are those most worthy of bibliographical treatment. In practice, some proposed (i.e. highly ranked) documents will be rejected in Phase 3, and some worthy ones may have been ranked so low in Phase 2 that the librarian never sees them (because available manpower will never allow the entire list to be handled).

The quality of Phase 2 ranking is essential for efficient use of manpower resources. All Phase 3 rejections may be logged, to be used as input to Phase 2 customization, though user functions/scripts. Customization facilities are identical to the Phase 1 ones; the only difference is the information available in the document vector.

If any catalog data is available for the document, e.g. because an earlier version was catalogued, or the title/author or ISBN was found in the document and information could be retrieved from a catalog, these data may be suggested as defaults, to be modified or not.

The librarian may modify the definition of the document extent. E.g. if the web page displays "Next" and "Previous" buttons, or displays a Table of contents where the entries are links to other web pages, the librarian may navigate to these pages and, if belonging to the same document, select "Include" in the user interface. The displayed pages are those archived in the DSM.

Catalog data for a dynamic network document may be invalidated by later document updates. The librarian may indicate that catalog data should apply to all future updates, to updates within a certain time (say, one year), or to this version only. He may also indicate that modifications beyond a certain threshold should trigger an exception, requesting a librarian to verify/update the catalog data. The amount of modification is determined by the size of the report from a **diff** comparison of the catalogued version and the revised one.

6 User Access

6.1 Access Control

Collections and Tickets. We propose that any document is stored in one or more *collections*. Documents in a given collection share access control, requiring a specific *ticket*, granted by a ticket service that is external to the DSM. To prevent a user from generating a false ticket, it is protected using cryptographic methods. A ticket has a limited lifetime, after which it must be renewed. It gives access to a specific set of operations – e.g., printing/saving may be prohibited. Tickets may be issued by e.g. rights holders external to the National Library, and may require payment.

Encryption. Legal deposit documents are available only for "research and documentation" purposes. To allow these user groups access without having to travel to the National Library, we propose that documents can be transferred to a local library or to the user's PC in encrypted form, interpretable only by a web browser plugin provided by the National Library. The plugin provides no printing or saving functions; temporary screen display is the only way to access it. The encryption key will be different in each session – an intercepted and saved data stream cannot later be displayed. Access will be restricted to users with a valid ticket; this is controlled through the ordinary access control mechanism. It is not yet clear whether this is sufficient for legal acceptance as an access method to the deposit library.

6.2 User Access Tools

A user may access the archive using an ordinary web browser. The web server developed in the Nordic Web Archive (NWA) uses an HTML *FRAME* structure to display the archive document in the lower frame. The upper frame contains a time line where each new version of a Dynamic document is marked; the user can walk through the document history by clicking the marks. There are also fields for searching and other functions. Navigation is available in the lower frame. When a link is followed to another document, the version that is retrieved is the one valid at the time when the referencing document was harvested; the user navigates in a "virtual past".

We consider the NWA web server an excellent foundation for archive access. Some functional extensions to the upper frame may be desired, such as an option to search in a library catalog, but the main structure is well suited.

7 Conclusion and further work

A complete architecture for the handling of network documents from harvesting to user access is a necessary foundation for creating a truly useful library of Internet resources.

An implementation of this architecture will to a significant degree be based on adaptation of existing software modules: Currently, the harvester software from the Nedlib project is investigated; a second alternative is the harvester currently being developed by the International Internet Preservation Consortium. User, librarian and technician access to the harvester database may be based on the Nordic Web Archive (NWA) software. The central Digital Storage Vault is currently in the implementation phase. Standard modules may be used for web proxy, NetNews and email handling. Adapting and integrating these into the architecture does not present major challenges.

The challenging part is developing the logic to analyze, group and automatically rank the collected documents, to ensure that manpower is not wasted on operations that could be automated, and resources focused on the appropriate fraction of the collection. Due to our total-harvesting approach, this is not time critical: As long as all available information is being saved in the harvester database, selection/ranking procedures may be gradually refined over time. The first major work step is therefore to integrate existing modules according to this architecture, after which we will attack the selection/ranking tasks. The Project Steering Committee takes the final decision about further work.

---

1 Networked European Digital LIBrary, http://www.kb.nl/coop/nedlib/

2 Home page: http://hosted.ukoln.ac.uk/biblink/

3 Home page: http://nwa.nb.no

4 Home page: http://www.nb.no/paradigma/eng_index.html

5 INDOREG 2 Registrering af statiske og dynamiske publikationer samt elektroniske tidsskriftartikler. Prosjektrapport. Dansk BiblioteksCenter 1999. ISBN 87-552-2424-5

6 Home page: http://www.greenstone.org/english/home.html

7 http://www.archive.org

8 Home page, English version: http://www.kb.se/kw3/ENG/Default.htm

9 Identification issues are addressed in detail in *Nuys and Albertsen: Identification of Network Accessible Documents: Problem Areas and Suggested Solutions*, presented at the 3rd Workshop on Web Archives

10 FRBR report, PDF format: http://www.ifla.org/VII/s13/frbr/frbr.pdf