Portable Full-text Retrieval System

Takehiko Murakawa Tatsuya Takehara
Faculty of Systems Engineering
Wakayama University
Wakayama, Japan
e-mail: takehiko@sys.wakayama-u.ac.jp

***Abstract*—We developed a portable full-text search system that runs on Microsoft Windows. The server programs, full-text search executables, and the PHP system files can be stored on a removable memory device, such as a USB flash drive. The storage location of documents and the index are assignable to the host PC or the USB drive. A browser-based crossover search can be performed on plain text files, Word and PDF documents, and Excel spreadsheets. While the system employs established executable files for the server programs and the full-text search engine, we developed interfaces to support system use. We tested the system with more than tens of thousands of records to verify immediate retrieval.**

Keywords-Web application development; full-text search; document retrieval; mobility

I. INTRODUCTION

The most desirable environment for document retrieval is one that allows quick and easy file retrieval. In addition, some users require the privacy, i.e., the protection of files, and some desire portable document retrieval to use in internet-incapable locations.

Personal full-text search cannot be realized exclusively by advancing internet technologies. For example, the field of humanities, many researchers find internet resources insufficient. However, they do retain text data in spreadsheets or document files to read later. These researchers genuinely require simple and functional full-text search and retrieval.

To satisfy mobility requirements, we have been developing a portable full-text retrieval service. In this paper, we describe a file retrieval system that is stored on a universal serial bus (USB) flash drive.

The system can function on a low-performance personal computer (PC) such as a netbook or laptop, running Windows XP or higher operating system (OS). The portable edition of XAMPP server programs and Hyper Estraier search engine reside on the USB flash drive. The registered documents are indexed by n-grams; therefore the search is language independent. Even though a file may contain multiple languages, the relevant documents can be located if an appropriate search term is used. Search results are presented as hyperlinks to documents that include the keyword. The system allows crossover searches of plain text files, Microsoft Word documents, Excel spreadsheets, and Portable Document Format (PDF) files.

The proposed system is an enhanced version of previously developed system [1]. While the fundamental features, such as file registration and retrieval, are consistent with those of the previous system, the proposed system employs portable executable files, including an open source Hypertext Transfer Protocol (HTTP) server, and MySQL, a database management system (DBMS). With careful selection of hardware and software and refinement of existing user interfaces, the service can be easily used on familiar Windows PCs.

The remainder of this paper is organized as follows. Section II explores the background of the full-text retrieval service. Section III details the services and software related to our system. Section IV describes the main body of the system, including hardware and software configuration, and provides details about several key features. Section V presents an evaluation of the system. Section VI acknowledges challenges and provides a general discussion to identify the significance of the system. Finally, Section VII presents conclusions and future work.

II. PRELIMINARIES

This section introduces the execution environment and defines key components of full-text search.

A. Execution Environment

We assume that humanities research consists of fieldwork and laboratory investigations. Researchers often visit temples, museums, or archive warehouses to examine unpublished historical material. Since some of the material might be unidentified, the researchers examine it with reference to known information. We have been developing support systems, primarily Web applications, for humanities researchers [1] [2] [3]. Currently, our system is aimed at full-text search of data and spot notations for subsequent full-text searches.

The data stored and searched in our system have some common characteristics, and researchers have common expectations. The content to be searched is usually in plain text format. Although researchers do encounter data in rich document formats, such as Word or PDF, they do not expect mechanical retrieval of figures or complicated structures. Since the researchers are interested in historical material that describes the human environments and events that occurred approximately 1,000 years ago in Japan, the documents are primarily written using Chinese characters. In addition, the users may wish to add a notation in Japanese to the material they locate and will expect to locate this material by searching for the notation. Our system targets researchers who are adept at using PCs and who understand what sort of

 ISBN: 978-1-61208-280-6

data is suitable for computer processing. The researchers' basic tasks are (1) to read the freshly discovered documents, (2) to add one or more notations to the document to be used by our retrieval service, (3) to find and read relevant documents, and (4) to estimate the age and/or verify the validity of the document.

B. Full-text Search

In this section, we define several common terms specific to the full-text search described in this paper. We define a *search engine* as software that performs a full-text search. The process that involves a search engine and search interfaces is called a *retrieval service*. Google and Yahoo! are global retrieval services for internet resources. Note that we are not attempting to develop a search engine for global or domain-specific use. Rather our goal is to provide a portable and user-friendly framework for retrieval services by using an existing search engine. The term *system* is equivalent to service, but is used when we are emphasizing internal structure.

The data a user inputs to initiate retrieval are referred to as a *search term*. A search term may be two or more words separated by a space. The smallest unit of information to be searched is called a *record*. The scale of the service is often represented by the number of registered records. We use *document* as a synonym of record.

Despite the existence and wide adoption of internet retrieval services, such as Google, retrieval services are receiving increasing attention. Local services have significant advantages; the most promising benefit of a local full-text search system is remaining confidentiality of in-house documents and documents where intellectual property rights are in dispute. Another advantage is the ability to recognize and increase the value of the material through Web mining and access log analysis [4] [5].

Hyper Estraier, adopted in our service, is a nonresident search engine. The service invokes an executable file every time it registers a record or sends a search query. We have ensured that the service developed previously worked well. Another sophisticated search engine is Apache Lucene [6], which is written entirely in Java.

III. Related Works and Software

A number of researchers have attempted to develop digital library retrieval services for documents written using Latin characters [7] [8]. In addition, some researchers have proposed systems for other writing systems. Chen, Hsiang, Tu, and Wu [9] reported a full-text search system for Taiwanese historical documents written in Chinese. The documents were housed in the National Taiwan University Library. Alshuhri [10] attempted to index and retrieve Arabic manuscripts stored in digital libraries. Batjargal, Khaltarkhuu, Kimura, and Maeda [11] focused on traditional Mongolian scripts and provided a query method using modern Mongolian. The authors have also experimented with a database of Chinese characters in Buddhist canons [3].

There are several methods, other than search engines, to retrieve documents. We will describe these briefly and examine their disadvantages. Grep is a well-known sequential search command originally developed for Unix environments, which is now available in Windows as well. Typically, grep does not create or readily use an index. Consequently, grep retrieval time is directly proportional to the volume of content searched. In addition, the file access (open, readout, and close) time would be a consideration. For these reasons, a grep-based approach to searching vast numbers of documents would be impractical.

It is difficult to perform a multiword search on large plain text files or large Excel spreadsheets. For example, it is very hard to find documents in which two words appear in no particular order. Moreover, AND/OR/NOT searches might be ineffective on such large files. The proposed method is intended to be suitable for text search of many small files and/or a few very large files. The problem of word order can be resolved using a search engine.

Recently, with the popularization of cloud computing, document management systems are being used more frequently. Evernote, a leading service, manages user records with tags called notebooks. Evernote and other popular services can operate across multiple devices such as PCs and smartphones. However, there are some practical problems associated with these services; large volumes of data may take a long time to upload and there are terms-of-use restrictions. Another reason for researchers to hesitate to use these services is the psychological resistance to the commission of their own data.

Before search engines matured, retrieval systems were constructed using DBMS, and queries using Structured Query Language (SQL) were executed for retrieval. It is true that some DBMSs included full-text search features, but they were not suitable for searching text data written in Japanese or Chinese. In addition, these DBMS-based retrieval systems do not handle records containing multiple languages effectively. For example, the SQL Server full-text search function that runs on Windows requires a specific language to be identified to generate the index.

In general, when constructing a retrieval service based on a DBMS, it is necessary to design the database carefully in advance, considering the content and search parameters. Therefore such a retrieval service is both restricted and inconvenient. DBMS-based services are suitable for narrow data retrieval, such as dates or numbers, but not for full-text searches.

IV. The System

Before describing the main body of the system, we clarify the requirements of our portable full-text search system. The search capability requires immediate and complete retrieval without communication with other computers. Note that search noise or false positive results can be corrected by better search terms. The system must be user friendly; users can perform and control a search with their own hardware. It is desirable to require minimal steps to facilitate a search. While the system described in this paper is compatible to the previous system [1] with regard to search capability, the notable feature of the proposed system is the use of a USB flash drive with any computer to perform full-text searches.

 ISBN: 978-1-61208-280-6

It is assumed that a system user is neither familiar with nor interested in the internal structure of the system. However, he/she can perform basic operations such as Web access and installing software. With regard to ease of installation, a retrieval service using virtualization, such as VMware or VirtualBox, is not a good solution. The installation and management of a virtual OS is complicated and potentially expensive. In addition, the software might perform poorly on portable, low-performance PCs.

Even though retrieval services such as those used with iPod [12] and electronic dictionaries are attracting or extensible, we excluded these implementations because they would be ineffective for handling record registration.

A. Software Configuration

Fig. 1 shows the overview of our system. The system is stored on a USB flash drive, and includes the server programs, the full-text search executable, the administrative files for the database, the source files for system operation, and the index and document files.

The server programs used in this system are from a portable edition of XAMPP server stack. We used the Apache web server, MySQL database, and Hypertext Preprocessor (PHP) interpreter. The full-text search executable, "estcmd.exe," performs retrieval on demand. The administrative file for the database maintains the search term conversion table, the list of keywords used in notification, and the bookmarks. The system does not require user authentication.

The system source files consist of Hypertext Markup Language files and PHP files based on Smarty, a PHP template engine. System directories are arranged according to Smarty conventions, and the source files are located in respective directories. We employed Cascading Style Sheets to manipulate the appearance of the pages and JavaScript to run client-side scripts. Windows batch files were used to activate and terminate the system.

The index is read or updated by the full-text search executable during registration and retrieval. The document files are linked on the retrieval results page for easy access. The user can also open the files immediately using the directory hierarchy of the flash drive.

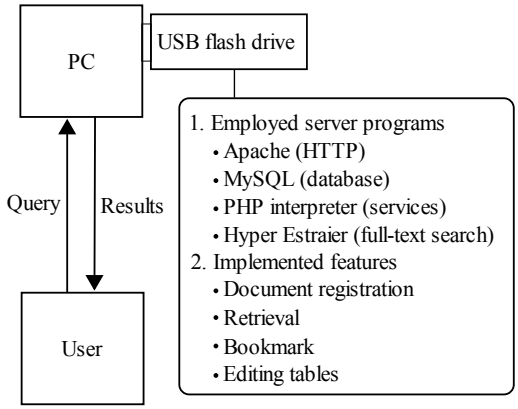

Figure 1. System overview.

B. System Setup

Fig. 2 shows the system setup and use workflows including USB flash setup.

To use the system, the USB flash drive must be prepared beforehand. To do so, we first plug the drive into a PC and install XAMPP. Then, we put Hyper Estraier executable and the system source files in place.

The database must be initialized during setup. When the batch file is run, it resolves the USB device drive letter and invokes XAMPP. We can then find the control panels and launch MySQL. We do not have to activate Apache at this time. The database is initialized by SQL setup queries.

To use the prepared drive with another PC, we stop the MySQL process, terminate XAMPP, and detach the flash drive.

The service begins by plugging the device into a PC and executing the batch file described above. Here we activate Apache and MySQL using the XAMPP control panel. When opening the Uniform Resource Locator as prescribed, including "localhost," we can see the service's front page (Fig. 3). Each page has links to retrieval ("[Search]"), document registration ("[Register]"), bookmarks catalog ("[Bookmark]"), conversion table editing ("[Term+]"), keyword editing ("[Keyword]"), and general service information ("[About]").

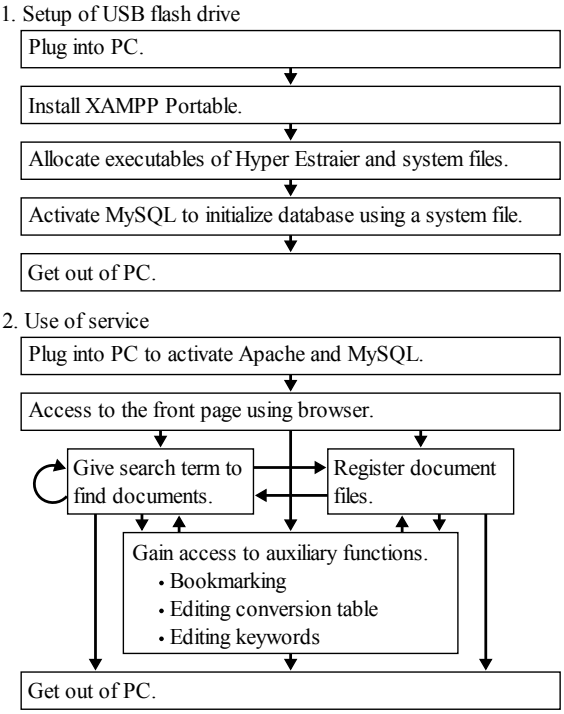

Figure 2. Setup and use workflows.

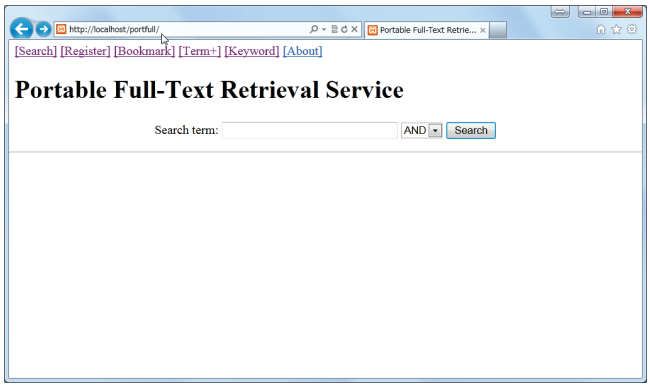

Figure 3. Front page of the service.

C. Registration of Documents

We also developed interfaces to support document registration (Fig. 4).

After opening the registration page, the user specifies the name of a folder and clicks the "Update index" button. Then, the page goes dark, and an "under construction" message appears at the center until registration finishes. At this stage, the user cannot interact with the page.

During this time, the system searches for files that should be registered by scanning the folder specified by the user. The text data are extracted from each file, and the search engine updates the index according to the data and file name.

The system accepts "txt," "doc," "docx," "pdf," "xls," and "xlsx" file extensions. There are three ways to preprocess the files for indexing. The text data of a "txt" file can be obtained through file loading. For a "doc," "docx," or "pdf" file, the batch file "estxfilt.bat" invokes "xdoc2txt.exe" to extract the content. Note that estxfilt.bat and xdoc2txt.exe are Hyper Estraier executable files. For "xls" or "xlsx" files, xdoc2txt.exe is called by the PHP interpreter to extract the data. Each row of a spreadsheet is regarded as a record to be registered. This means that an Excel file is generally split into a number of records for the indexer. This feature was derived from a request by humanities researchers who used a prototype service.

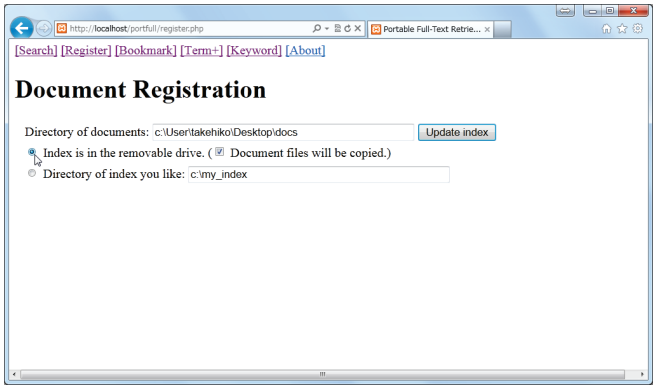

Figure 4. Front page of the service.

D. Retrieval

Fig. 5 shows the retrieval results where "1234" was given as the search term after plain text files were registered.

The screen displays the search time and the number of relevant and total records. The relevant records are shown in a table. The table also includes hyperlink to the relevant document and a snippet, i.e., content from the document around the search term. In the snippets, when the search term is highlighted, the keywords are hyperlinks. When one of these hyperlinks is clicked, a search using this keyword begins.

When many records match the search term, the table only displays the top 10 records. A "[More]" hyperlink hidden under the table is available to display the next 10 records. This visualization method avoids pagination and the page transitions. Note that search results can be reduced and improved by adding an extra word to the search term.

E. Other features

Our system supplies three reusability features, i.e., bookmarking, conversion table editing, and keyword editing.

We implemented a unique bookmark feature to allow the user to store the desired results. Note that this feature differs significantly from browser's bookmark feature; a traditional bookmark is associated with a page corresponding to a search result or a record. In contrast, the implemented bookmark module manages and displays a suite of records that is configurable by the user.

Bookmarking begins with the search result. The user checks one or more records and then clicks the bookmark button on the left of the table. The user is then presented with a new page.

The new page is composed of forms to assign bookmark names, associated notes, and links to the stored records. When the user fills in the forms and presses the registration button, the set of bookmarks is stored in the database. These bookmarks can be seen at any time. When the user selects a bookmark from the list in the top menu, the bookmark page, including the name, notes, and links to the records, is displayed.

Figure 5. Retrieval results.

Prior to describing the search term conversion table editing feature, an explanation of the search term conversion in the previous system, which was intended to support humanities researchers, is required. When a user specifies "1234" as the search term in the previous and proposed systems, the service identifies several Chinese strings, each of which means the year 1234 in Japanese. The actual Hyper Estraier query is performed by disjunctive search, where the search term consists of the original search term and derived words with the pipe symbol ("|"), i.e., the OR operator, inserted. The resulting records include at least one divisional search term.

To date, data for search term conversion have been extracted primarily from documents obtained from humanities researchers. Taking into consideration the variety of users and use cases, we must satisfy the requirement to add, modify, and remove conversion rules. To address this, we implemented an interface for editing the search term conversion table using PHP.

When the user clicks "[Term+]" to open the configuration page, he/she must first specify the candidate search term to be converted. Then he/she must click the search button. If the term has already been registered, or one or more ex-post terms exist, then the table displays all available patterns (Fig. 6), and the user can modify and/or delete these items. If the term was not registered, then the user can fill in an ex-post term and click the button for registration. Since there are currently only a few rules, the system attempts to match search term conversion to an exhaustive search of the conversion table, which is managed by the DBMS.

In a similar manner, we also provided an interface for editing keywords displayed in the search results.

V. System Evaluation

We conducted experiments to verify the usefulness of our service. Throughout experiments, we used XAMPP Portable Lite 1.8.1 (including Apache 2.4.2, MySQL 5.5.27, and PHP 5.4.7) and Hyper Estraier for Windows 1.4.10. These program files were put on a USB 3.0 flash drive.

First, we registered over 12,000 text files that were used in an evaluation of the previous system. The text files and index were stored on the PC but not on the USB flash drive.

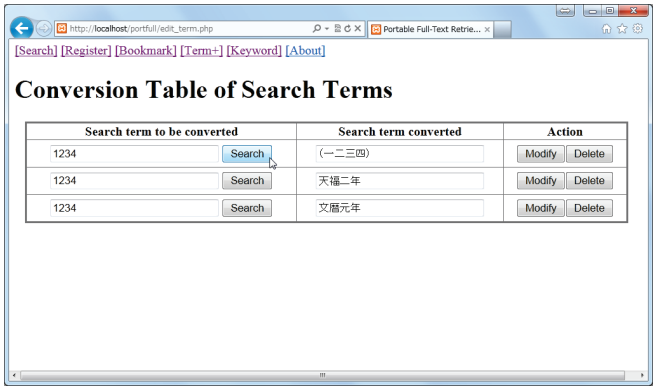

Figure 6. Editing search term conversion table.

Although it took approximately 20 min to set up, we obtained immediate search results in each case. We also ensured that when we terminated the programs and unplugged the USB flash drive, no processes or files related to the service continued to run.

We also investigated the quantitative properties of the system through index construction and search, using three PCs and two datasets. One dataset consisted of 70 PDF files (1.2 GB total), which were scanned from research papers, reports, and books written in Japanese on humanities researches. The other dataset was composed of 14 Excel files (17 MB total) with bibliographical data on ancient Japanese documents, written in Chinese and Japanese. The aggregated number of records from the Excel files exceeded 44,000. All the contents and index were put on a USB flash drive.

Table I shows indexing and search times, and provides specifications of the three PCs. Each search time in the table is the average of search times, which Hyper Estraier reported, for retrieval using predetermined three search terms. The indexing time depends on the volume of records and PC performance, although the number of records did not seem to significantly impact the performance. The type of OS and PC capability also did not have significant impact on search time.

VI. Discussion

Through the experiments, we confirmed that the system can produce a result and list of files relevant to the search term in a short time, even with a low-performance laptop PC.

The major difference between the proposed system and our previous system is the storage location of the programs, index and content. The proposed system can preserve all files on a USB flash drive. Once file registration and indexing is performed on a high-performance PC, the user can put the flash drive in a low-performance laptop PC to enjoy effective search capabilities outside of a laboratory environment. We supplied a way to use the system taking usability and security into consideration; document files and the index file can be maintained on the PC, and the user may select the file location.

TABLE I. Indexing and Search Time

Hardware	PC #1	PC #2	PC #3
OS type	Windows 8	Windows 7	Windows XP
CPU	Core i7 2.00 GHz 4 threads	Core i7 2.80 GHz 4 threads	Atom 1.60 GHz 2 threads
RAM	8 GB	8 GB	1 GB
USB Support	3.0	3.0	2.0
Dataset #1 (PDF files)			
Indexing time	650 s	704 s	1,508 s
Search time	1.20 s	1.28 s	1.07 s
Dataset #2 (Excel files)			
Indexing time	123 s	98 s	286 s
Search time	1.25 s	1.31 s	1.16 s

 ISBN: 978-1-61208-280-6

We also investigated the security and convenience of the full-text service. The operation using a client-server model was exposed to eavesdropping and access by the general public, both of which are unwanted by the content holder. Such a system would have a serious defect if it is unavailable in internet-incapable environments. In contrast, stand-alone execution has no security or operability risks; however, the computer should be protected against theft, because even though we do not need to worry about communication issues when using our system, we must consider the fact that USB flash drives are easy to lose or steal. To address this problem, security will be enhanced by encrypting the data on the flash drive and/or by employing software authentication. However, the convenience of interoperability among different computers remains a significant benefit.

Taking these properties into account, we would like to highlight the usefulness of our service. For example, when a research project task requires that text data be produced from a series of historical material, it is not always efficient to introduce and maintain a server in the laboratory or the location in which the collection is stored. Such a solution would require hardware and software costs, and must have guaranteed communication infrastructure. Since our service is free from communication requirements, the user can interact with only the USB flash drive and their own PC. Therefore it is much easier to create a text management environment that includes referencing and uploading of data.

We can register and search documents written in Japanese, Chinese, or any other language using our system because Hyper Estraier accepts UTF-8 encoded text data. For a string found in the historical material with characters unregistered with Unicode, the character code may be found in Mojikyo and described using ASCII characters in the record. In that case, such a character will not be displayed in the browser, but could be visualized by rendering software not included in our system or may visualized by the researcher.

VII. CONCLUSION

We have reported a portable full-text retrieval service that does not require communication with other computers. A simple USB flash drive can retain all the executable files, system files, document files, and the index; therefore, users can easily enjoy the service using their favorite Windows PC.

In the future, we will conduct tests to ensure that the system functions in a practical environment. In addition, an examination of the system's scalability in relation to the number of registered files or total number of characters is also required.

ACKNOWLEDGMENT

The authors thank Prof. Keigo Utsunomiya for his advice relating to humanities research and the significance of personal or small-scale document management. They would also like to thank Enago (www.enago.jp) for the English language review.

REFERENCES

[1] T. Murakawa, Y. Watagami, K. Utsunomiya, and M. Nakagawa, "Full-text retrieval system for humanities researches," Proc. Tenth Joint Conference on Knowledge-Based Software Engineering (JCKBSE 2012), 2012, pp. 118–127.

[2] T. Tanaka, K. Fukimbara, K. Utsunomiya, H. Morikawa, and M. Nakagawa, "Database of Japanese Buddhists in the 10-13 centuries," Proc. 36th International Conference on Modelling and Simulation of Systems (MOSIS'02), 2002, pp. 265–272.

[3] T. Tanaka, Y. Nino, R. Zhang, M. Rolland, M. Nakagawa, S. Aoki, K. Utsunomiya, and T. Ochiai, "A database system of Buddhist canons," Proc. Seventh Joint Conference on Knowledge-Based Software Engineering (JCKBSE 2006), 2006, pp. 327–336.

[4] R. Cooley, B. Mobasher, and J. Srivastava, "Web mining: information and pattern discovery on the World Wide Web," Proc. 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'97), 1997, pp. 558–567.

[5] Z. Su, Q. Yang, H. Zhang, X. Xu, and Y. Hu, "Correlation-based document clustering using web logs," Proc. 34th Hawaii International Conference On System Sciences (HICSS-34), 2001, pp. 3–6.

[6] H. Li, W. Li, G. Wang, and X. Peng, "Information retrieval services based on Lucene architecture," Information Computing and Applications, Communications in Computer and Information Science, 2012, pp. 638–645.

[7] M. Gonçalves, "Digital libraries," Modern information retrieval (Second edition), Pearson Education, 2011, pp. 711–735.

[8] G. Buchanan, S. J. Cunningham, A. Blandford, J. Rimmer, and C. Warwick, "Information seeking by humanities scholars," Proc. 9th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2005), LNCS 3652, 2005, pp. 218–229.

[9] S.-P. Chen, J. Hsiang, H.-C. Tu, and M. Wu, "On building a full-text digital library of historical documents," Proc. 10th International Conference on Asian Digital Libraries (ICADL 2007), LNCS 4822, 2007, pp. 49–60.

[10] S. S. Alshuhri, "Arabic manuscripts in a digital library context," Proc. 11th International Conference on Asia-Pacific Digital Libraries (ICADL 2008), LNCS 5362, 2008, pp. 387–393.

[11] B. Batjargal, G. Khaltarkhuu, F. Kimura, and A. Maeda, "Ancient-to-modern information retrieval for digital collections of traditional Mongolian script," Proc. 12th International Conference on Asia-Pacific Digital Libraries (ICADL 2010), LNCS 6102, 2010, pp. 25–28.

[12] D. Bainbridge, S. Jones, S. McIntosh, I. H. Witten, and M. Jones, "Beyond the client-server model: self-contained portable digital libraries," Proc. 11th International Conference on Asia-Pacific Digital Libraries (ICADL 2008), LNCS 5362, 2008, pp. 294–303.

 ISBN: 978-1-61208-280-6