International Journal of Software Engineering and Its Applications
Vol.7, No.5 (2013), pp.337-354
http://dx.doi.org/10.14257/ijseia.2013.7.5.29

A Model Building of the Multi-Functional Digital Archives System for Traditional Knowledge

Hoon Jo[1], Hong-chan Jeon[1], Han-hee Ham[2] and Soon-cheol Park[1]

[1]*Division of Electronics and Information Engineering, Chonbuk National University, Jeonju, Jeollabuk-do, Korea*

[2]*Department of Archaeology and Cultural Anthropology, Chonbuk National University, Jeonju, Jeollabuk-do, Korea*

{ei201250226, ei200611816, hanheeh, scpark}@jbnu.ac.kr

Abstract

This paper proposes a solution for storing and conjugating the traditional knowledge archives system which draws international attention. The world's traditional knowledge is disappearing so fast that its documentation and archives should be considered urgently and significantly. We have created the traditional knowledge archives system based on the Web 2.0 platform to enhance the preservation of traditional knowledge. The major contributions of our system in the areas of both digital archives and traditional knowledge are twofold: an integrated archive structure using high technical mechanism and a multi-functional application for the public.

Keywords: *KTDAS, Digital Archives, Traditional Knowledge, Dublin Core, Semantic Search, Map Search, Detail Search, Chronological Table*

1. Introduction

Traditional knowledge is disappearing so fast due to the rapid process of modernization and industrialization. It draws attention to necessity and importance of archiving the traditional knowledge [1]. Examples of the traditional knowledge archives system are NDL (National Diet Library. Japan) [2], HOLOCAUST (United States Holocaust Memorial Museum)[3], and DENSHO(Densho) [4]. Although these systems have their own different type of functions and structures, their purposes are to preserve and apply the missing knowledge and materials. However, they have not overcome the limits of simple computerization and data services with archive materials.

The Korean Traditional knowledge Digital Archives system (KTDAS) suggested in this paper provides very efficient and effective approaches on the collection materials of the Korean people's lives in the twentieth century in view of time, place, and semantic meaning.

This paper is organized as follows. The next section discusses the structures and characteristics of KTDAS. Section 3 describes the functions of various search mechanism and the chronological table in KTDAS. Each search suggested by the system with characteristics of it is described in Section 3. Section 4 covers the implementation of KTDAS and the comparison with other system. Section 5 concludes and discusses future works.

ISSN: 1738-9984 IJSEIA

2. Structure of KTDAS

The structure of KTDAS is divided into the structure of the database and that of application based on database. The application manages data systemically and provides data to users with Web. The database makes it possible to offer data using a variety of methods with three more categories that are location information, GPS, and Row Data Place, which are based on the Dublin Core. KTDAS System classifies data that archivists collected with database of the extended Dublin. Also, it offers them to users with applications in different ways.

Table 1. KTDAS's Database Schema

	Elements	Type	Contents
1	Title	varchar(50)	Title of the resource to be described
2	Creator	varchar(40)	Name of the person or organization primarily responsible for creating the intellectual content
3	Subject	varchar(100)	Your own keywords describing the topic of the resource
4	Descriptions	medium text	Abstract, content description
5	Publisher	varchar(80)	University department, corporate entity etc.
6	Contributor	varchar(30)	Name of significant contributors other than the creator
7	Format	varchar(30)	Data representation of the resource
8	Date	varchar(15)	Date associated with the creation or availability of the resource
9	Type	varchar(30)	Category of the resource
10	Identifier	medium text	String or number used to uniquely identify the resource described by this metadata
11	Source	varchar(90)	Unique string or number for a printed or digital work from which this resource is derived
12	Language	varchar(10)	Language of the content of the resource described
13	Coverage	medium text	Spatial and/or temporal characteristics of the resource
14	Relation	medium text	Relationship to other resources
15	Rights	varchar(150)	Link to a copyright notice etc.
16	Raw Data Place	varchar(150)	Location to store the raw data.
17	RFID Tag Number	varchar(50)	RFID Tag Number
18	GPS ID	varchar(50)	GPS ID

The extended Dublin Core is selected for our KTKDAS metadata schema. Since data become increasingly diverse in forms and contents, the old metadata format has limits in terms of document centric description [5-6]. Table 1 represents elements of the extended Dublin Core from the existing Dublin Core [7]. The extended Dublin Core is that Raw Data Place, RFID Tag Number, and GPS ID are added to the existing fifteen elements. Raw Data Place indicates location data of raw data. RFIP Tag Number and GPS ID are the added elements to perform a range of functions.

Figure 1 illustrates the database diagrams according to the elements given on Table 1. The database is divided into four mechanisms. Metadata and data types are entered into tables of Digital Archives based on elements of extended Doublin Core. The 'archives_meta' table of Digital Archives became the basis of implementing additional functions. Then, a table of Digital Dictionary includes information for connecting related archive data with term data to compile a dictionary. The structure of those tables is to refer 'archives_meta' to connect archive data.

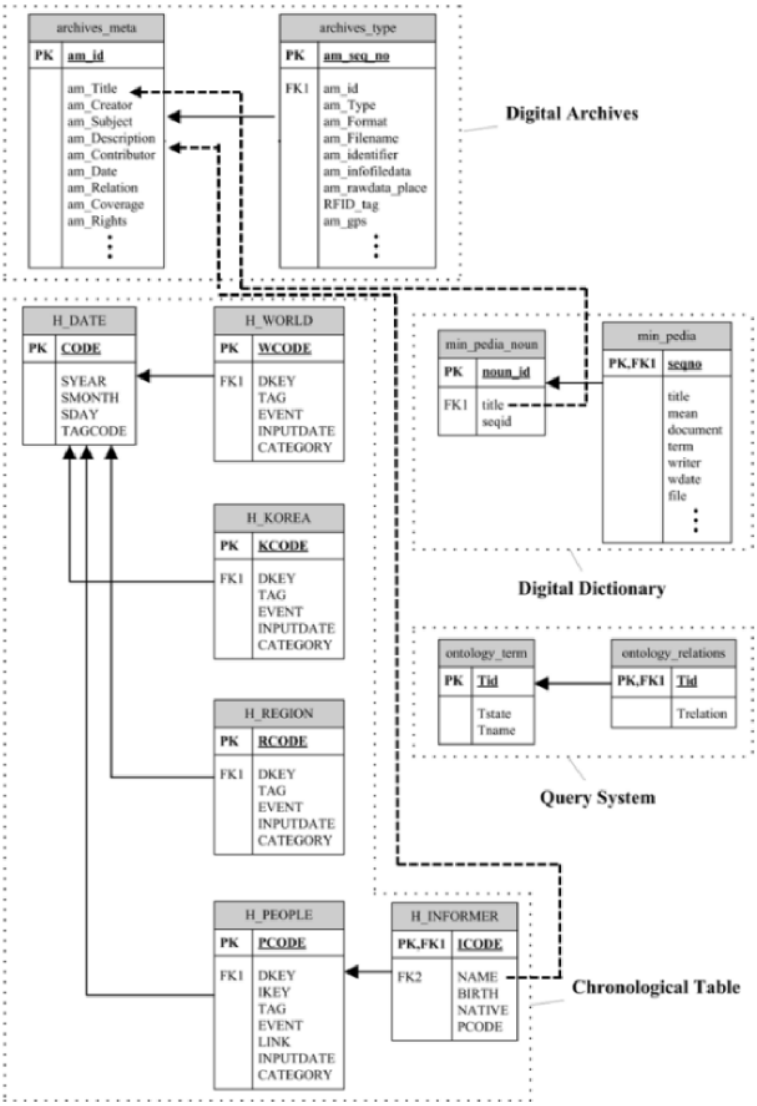

Figure 1. Database Tables Correlation Diagrams

General archives have aimed for compressing the size of data and retaining them permanently. However, KTDAS have aimed for effective search and utilizing data as well as compressing the size of data and retaining them permanently. In order to effectively implement, our KTDAS is designed for layered architecture based on database and indexed database.

Figure 2 shows the layered architecture used in the applications and how they relate to each other. Figure 2 shows the layered architecture for the KTDAS based on database. Data Access Layer processes a procedure that reduces the cost for searching the collected data. In this process, it creates an index out of the stored database by Lucene, thus providing faster search from the created index. Processed materials are offered to Interface Layer, which functions as a stage to give the processed materials to users. The function is finally processed to various applications for users, using Application Layer.

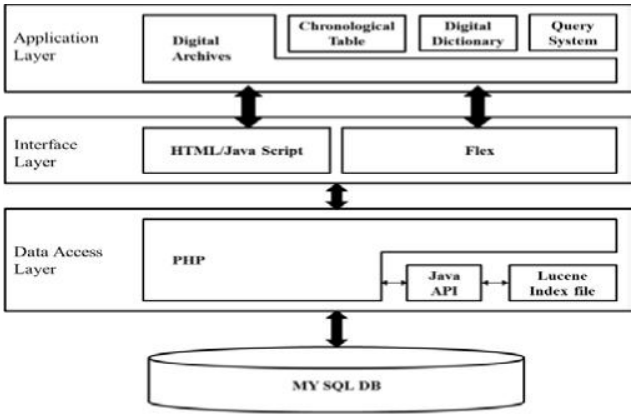

Figure 2. Layered Architecture for the KTDAS

3. Functions of KTDAS

Figure 3. Overall System View with its Applications

Based on the features of these functions, they are classified into classified into (a) Simply query search, (b) Semantic search, and (c) Search based on database and its application according to features of the fuctions. The detailed explanations are as follows.

The simply search based on Lucene, draws the highly accurate results by using 'Must' of BooleanQuery's. Also, the semantic search is executed based on Lucene, using automatic term networks generated by the archivist in our research group. Those searches have the advantage to process massive data in the shorter time. They support very effective search systems by means of morphological analysis.

The detail search query based on QBE(Query By Example) can be used easily filling the query windows by users who want more detail search. Since the map search needs the accurate location of the content, it is made up with database structure based on extended Dublin core. It proves that it allows users to search as to location data, not just search with simple keywords. Also, the Chronological table is conducted for searching

in terms of time. As the table provides world history and Korean history of the same period, users can search personal history based on the historical background and figure out how the historical background affects people.

3.1. Simply Query Search

In order to implement Simply Query Search, the open source search system of Lucene was used. Lucene is an advanced Information Retrieval (IR) Library which can be extended. It supports simple addition of index and search tool in various applications.

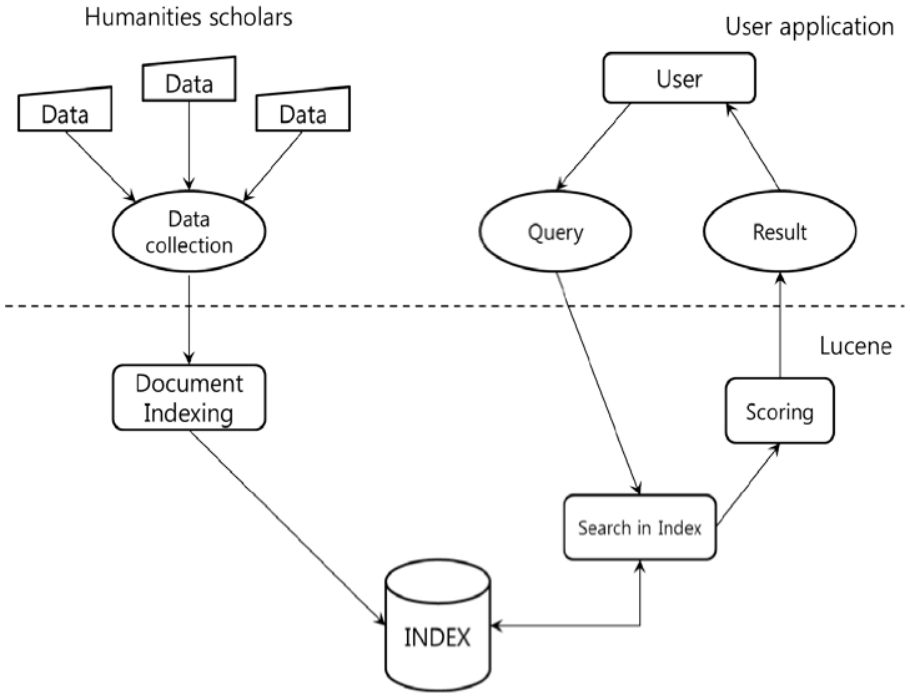

Figure 4. Processing for Indexing

As shown in Figure 4, this system stores data collected by archivists in accordance with Doublin Core. The stored data create the index through Document Indexing to be used for Lucene.

The Index is a special form of data storage utilized for quick searching, and it includes information on the documents in which each word is located. If request for search is made by users, it is switched to a Query to be applied in Lucene, and the search results are retrieved from the Index. The index goes through Scoring by a function ***Score*** of the document satisfying query, and then the scored document gets "Ranking". Consequently, users get the document of the highest ranking.

In order to carry out search requested by users, Query has to be generated. Prior to the generation of Query, however, the desired document has to be added in the Index. Lucene uses the Indexwriter Class for this process. To add the document in the Index via Indexwriter, the contents of the document need to be added in the form of a "Document Instance" in the "Field". The Field provides Keyword, UnIndexed, Unstored, and Text Field in Lucene accordingly to the purposes.

In case a user searches a document containing 'History' in a field of 'Contents' in a text format, Query can be formed into the following:

Term term = new Term("Contents", "History");

Query query = new TermQuery(term);

The example above is considered proper for a simple way of searching a term. For TermQuery, however, it is limited to complicated and qualified Query.

The system applied BoleanQuery to create proper query on users' requests. Boolean can choose from optional, essential, and restrictive characteristics and each of them can be explained with logical combination of OR, AND, and NOT [8].

Table 2. BooleanQuery in Lucene

Command	Enum Constant Summary
MUST (AND)	Use this operator for clauses that must appear in the matching documents.
SHOULD (OR)	Use this operator for clauses that must not appear in the matching documents.
MUST_NOT (NOT)	Use this operator for clauses that should appear in the matching documents.

Commands for the logical combination are on Table 2. BooleanQuery that consisted of these commands is as follows:

Extraction term from sentence to Term_Array

BooleanQuery booleanQuery = new BooleanQuery();

FOR i = 1 to N DO

Term term = new Term("Contents", Term_Array$_i$)

Query query = new TermQuery(term)

booleanQuery.add(query, BooleanClause.Occur.MUST)

END FOR

After Lucene extracts a term from the sentence of searching query, it proceeds to save the term in the Term Array. Group of stored terms uses 'Must' command to operate 'AND' with BooleanQuery, and then generates query.

Table 3. Example of Group of Indexed Documents

	Terms that occur in the document
Doc1	north, south, **war**, korea, **war**, prison
Doc2	people, life, **world**, **war**, japanese, prison
Doc3	history, life, marry, birth, born
Doc4	people, korea, **war**, life, **world**, **war**, japanese, prison

If a group of indexed documents are provided as shown in Table 3, and terms in each document are displayed, a searching query which contains the terms "world" and "war" can be carried out via BooleanQuery. Table 4 illustrates the search results of the BooleanQuery conducted under different commands such as "MUST", "SHOULD", "MUST_NOT".

Table 4. Executive results with "world" and "war" by BooleanQuery

Condition		Documents			
world	**War**	Doc1	**Doc2**	**Doc3**	**Doc4**
MUST	MUST	false	true	false	true
MUST	SHOULD	false	true	true	false
MUST	MUST_NOT	false	false	false	false
SHOULD	MUST	true	true	false	true
SHOULD	SHOULD	true	true	false	ture
SHOULD	MUST_NOT	false	false	false	false
MUST_NOT	MUST	true	false	true	false
MUST_NOT	SHOULD	true	false	true	false
MUST_NOT	MUST_NOT	false	false	true	false

BooleanQuery using 'MUST' command selected more accurate results with terms "world" and "war", which are Doc2 and Doc4. A function of Score determined the ranking of results of the searched documents. A function Score of query 'q' and document 'd' defines as follows.

$$score(q, d) = EtEq\ tf(t\ E\ d) * idf(t) * getBoost(t\ E\ q) * getBoost(t.\ field\ E\ d) * lengthNorm(t.\ field\ E * coord(q. * queryNorm(sumOfSquaredWeights) \quad (1)$$

tf(t∈d) means frequency, indicating how many times the term 't' is included in the document 'd'. As idf(t) is the inverse number of frequency in indexing a inverted file, the more documents that include the term, the more common the term is considered.

$$lengthNorm(t.field\ E\ d) = \frac{1}{4FieldTerrits} * f.getBoost * d.getBoost \quad (2)$$

Coord is the output of the number of the term in query from the searched document. Boost(t.field∈d) represents weight according to fields adjusted during indexing.

lengthNorm(t.field∈d) is stored by calculating while indexing as a output normalizing the number of terms included in Field and it defines as follows.

LengthNorm is a figure for calculating the importance of the term. The smaller the number of the term is in that field, the better figure we get. FieldTerms denotes the number of the term in the field; f.getBoost represents weight of the field; d.getBoost represents weight of the document.

$$queryNorm(somOfSquaredWeights) = \frac{1.0}{sumOfSquaredWeights} \quad (3)$$

$$sumOfSquaredWeights = \sum_{tEq} (idf(t) * getBoost(t\ E\ q))^2 \quad (4)$$

queryNorm(sumOfSquaredWeights) in equation(3) is outcome from normalizing the sum of the squared weights. sumOfSquaredWeights in equation (3) is equation (4).

Table 5. Score About 'War'

	tf	idf	getBoost		length Norm	coord	query Norm	score
			t∈q	t.field∈d				
Doc1	2	1	1	1	0.40	1	1	0.8
Doc2	1	1	1	1	0.40	1	1	0.4
Doc3	0	1	1	1	0.44	1	1	0
Doc4	2	1	1	1	0.35	1	1	0.7

Drawing from the Rankings gained through Score, Doc1 records the highest rank. In this study, the document which had gone through the Lucene's indexing procedures, noted previously, was deduced its Ranking via the function of Score. Similar documents were retrieved using the BooleanQuery and its 'MUST' command to deduce this Ranking. It offers better search method with morphological analysis, not just from String Mach.

3.2. Semantic Search based on a Term Network

In order to implement semantic search suggested in this report, we created a term network that represents ontology-based term relations. Ontology system is the main function for semantic Web application, which is a type of a glossary made up of terms and their relationships.

The system calls Ontology system a term network. A term network is built as archives data.

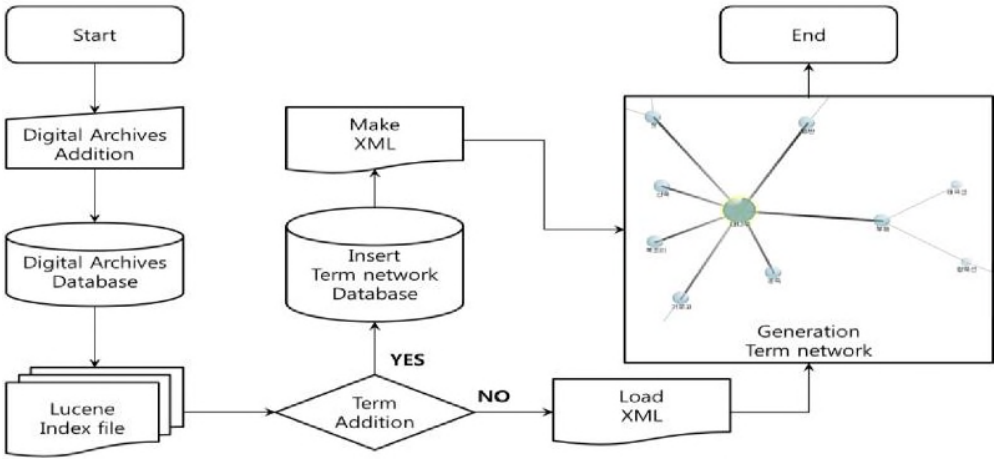

Figure 5. Automatic Formation Process of Term Network

A term network is divided into automatic generation and archives' entering according to the generation processes [9]. At first, Figure 5 is an automatic formation process of term network. Since data are entered, those are added into archives database. Then, files are processed to index by Lucene. If a term network procedure senses the addition of terms, it finds relative materials in Lucene index, makes XML documents that show relationship between terms and establish a term network. However, when there are no additional terms, it uses the existing XML documents and builds a term network.

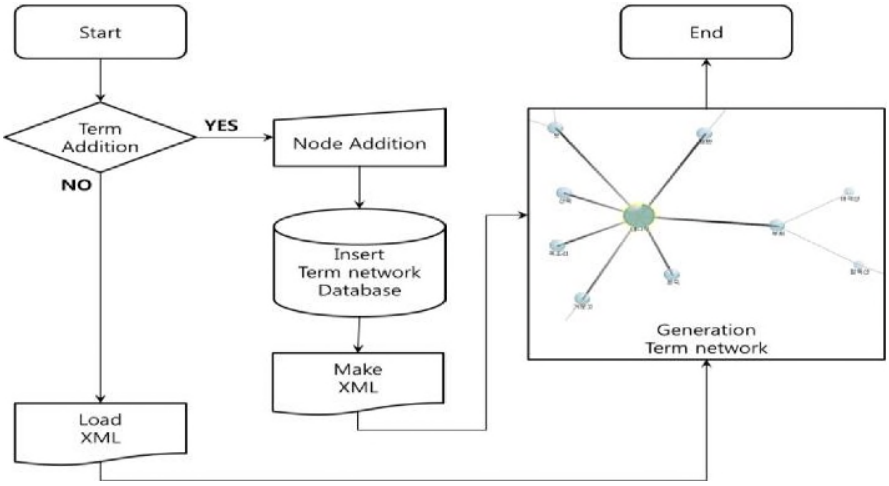

Figure 6. Process of Generating a Term Network by Archivist's Input

Figure 6 shows the process of generating a term network by archivist's input. Archivists choose a node of the formed term network, followed by the addition and deletion function of node. If it proceeds to addition by archivists, the term is added to the subordinate node of the chosen term. After that, term network procedure build relationship between new terms, make XML, and generate a term network. If there are no additional terms, it loads XML to generate a term network.

The Semantic search that the report suggests automatically drags terms from the generated a term network, and it creates the extended query system by applying weighted values. Figure 7 describes algorithm of the extended query search system. Once users' query is entered into the search system, the system performs searching, extending query by ontology depending on whether it uses the extended query search. Extension of query is conducted by extracting terms from ontology, figuring out the relations between the terms, and calculating weight.

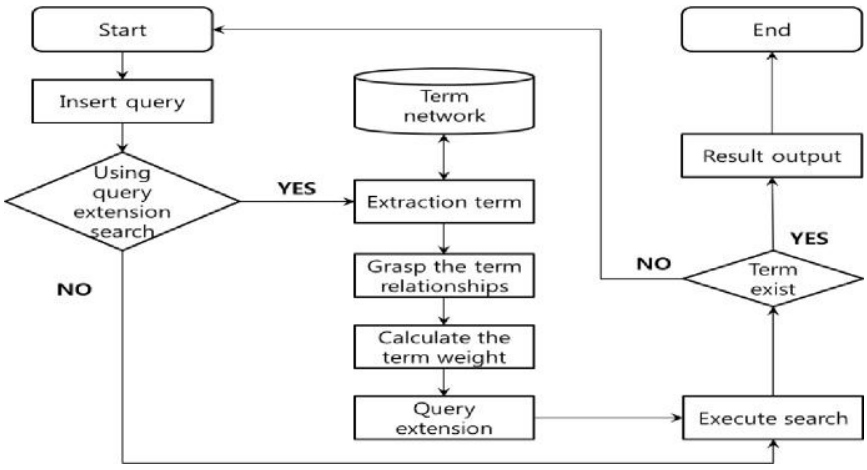

Figure 7. Flowchart of Extended Query Search System

It uses a calculating algorithm to calculate weight of the term. For the relationship between terms, it computes relations between two terms from WordNet and applies equation (5).

$$f(t_i, t_j) = f_{t_i,t_j}(l) \times f_{t_i,t_j}(h) \quad \textbf{(5)}$$

$$f_{t_i,t_j}(l) = e^{-\alpha l} \quad \textbf{(6)}$$

$$f_{t_i,t_j}(h) = \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} \quad \textbf{(7)}$$

In equation (5), f_{t_i,t_j} represents the relationship between the terms t_i and t_j. Equation (5) is further described by equation (6), indicating that if the distance l between t_i and t_j is bigger their relationship is smaller, and equation (7), showing that the larger the height h of the parental term from which the t_i and t_j had stemmed from, the larger is their relationship [10]. α, β in equation (4) and (5) are arbitrary constants and are the most effective when each of their values are 0.2 and 0.6 respectively.

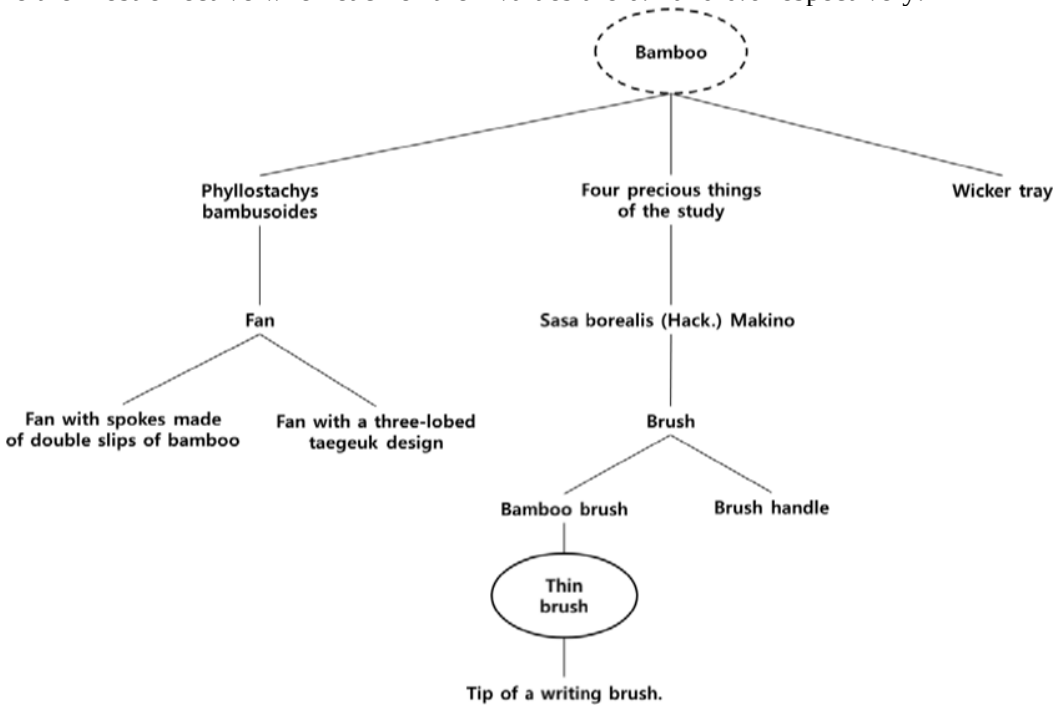

Figure 8. Term Network about 'Bamboo'

Figure 8 is a term network of terms related to “Bamboo”. The value h of the term “Bamboo” at the top node is 1, and the value increases as terms descend farther from the top node. The common value of h for terms “Wicker tray” and “Thin Brush” is 1, and the distance l, a value of 6, can be measured from the middle nodes, including their own nodes: “Wicker tray” – “Four precious things of the study” – “Sasa borealis (Hack.) Makino” – “Brush” – “Bamboo brush” – “Thin brush”. Table 6 shows the values h and l of each node and $f(t_i, t_j)$ which is calculated from the two values.

Table 6. Relationship Value of Each Term

t_i, t_j	H	l	$f(t_i, t_j)$
Wicker tray, Thin Brush	1	6	0.16
Fan with spokes made of double slips of bamboo, Bamboo brush	1	7	0.13
Tip of a writing brush, Brush handle	4	4	0.44
Fan, Sasa borealis (Hack.) Makino	1	4	0.24
Fan with spokes made of double slips of bamboo, Fan with a three-lobed taegeuk design	3	2	0.63
Fan, Wicker tray	1	5	0.19
Brush handle, Sasa borealis (Hack.) Makino	3	2	0.63
...	...	...	...

$$T = \{t1, t2, \ldots, ti\} \quad \textbf{(8)}$$

$$W = \{w1, w2, \ldots, wi\} \quad \textbf{(9)}$$

$$\text{Query}' = \sum_{i=1}^{n} (ti \wedge wi) \quad \textbf{(10)}$$

The method of query extension is to use T, a set of extended terms by ontology in equation (8), and W, a set of weight in equation (9), thus extending inserted query. equation (10) is reset extended term query for applying inserted query to Lucene.

Lucene links t and w with *caret* (*A*). In order to compute the Ranking search result of the searched documents, Lucene uses the function of Score like equation (1)

Figure 9 shows a term network from the process above and Semantic search from Ranking.

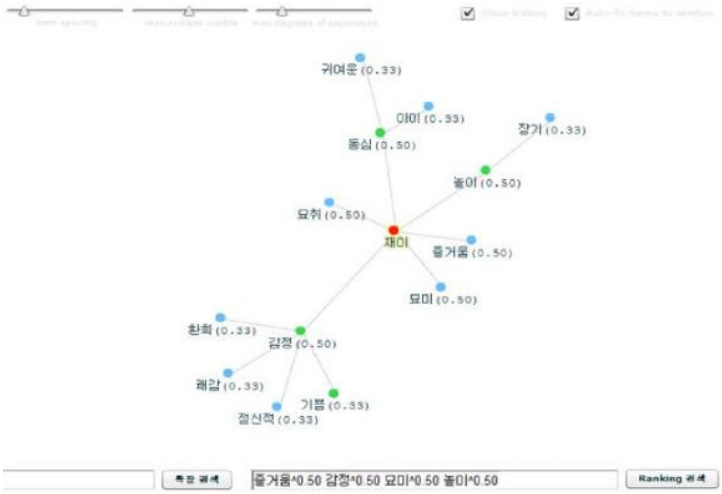

Figure 9. Semantic Search based on a Term Network

Figure 9 is the screen that shows Semantic search by the system in a tree structure of GUI based on Flex. It has an advantage in letting users access the related terms with the

Interface above and checking the similar terms with the main terms. In addition, we can search the related terms and their contents by clicking the node [11]. It makes it possible to approach semantically, not just simple String search.

3. 3. Search based on Database and its Application

3. 3. 1. Detail Search based on Database

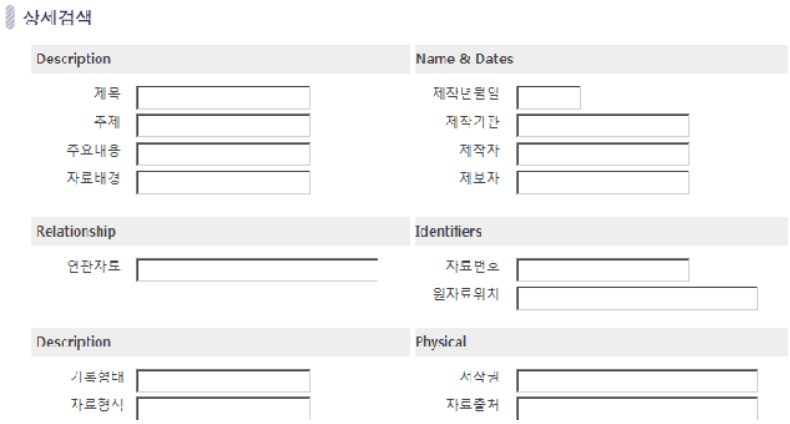

Figure 10. Advanced Search

Figure 10 shows Advanced search. Advanced search follows the way for users to voluntarily input conditions on each field, which is Query by Example (QBE).

QBE is a language for querying in order to search and modify data in relational database management system (RDBMS). It allows users to search data from database by simply setting conditions in a table on the screen (Wikipedia).

It has an advantage in setting conditions easily for users who don't have enough knowledge about query. The system took advantage of it and created the advanced search, by providing an easier way to use on the complex query of Meta Table about meta information and Type Table about type information of data.

3. 3. 2. Chronological Table

The system embodies history of personal life from people's collected life and provides lives of the common people who went through the twentieth century of the turbulence in chronological order. Furthermore, it indicates how world history and Korean history affect a personal life. It allows users for chronological search beyond the simple keyword search. Here are the details.

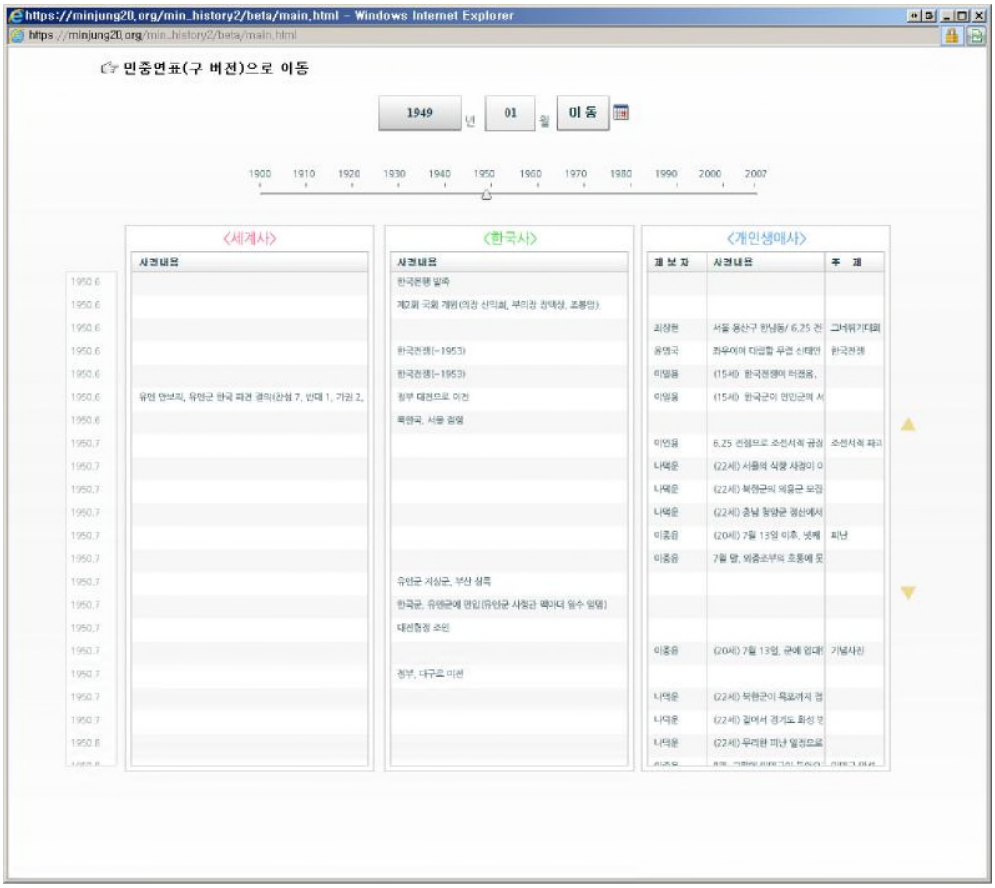

Figure 11. Chronological Table

Chronological table is the system that supports search with respect to world history, Korean history, and individual life history. Users can suspect the background of events chronologically on how historical events affect individual life history. For example, contents of the 1950s are mainly about refuge, military force, and food. This is because it was affected by the Korean War that began in July 1950 and UN forces dispatched by United Nation Security Council.

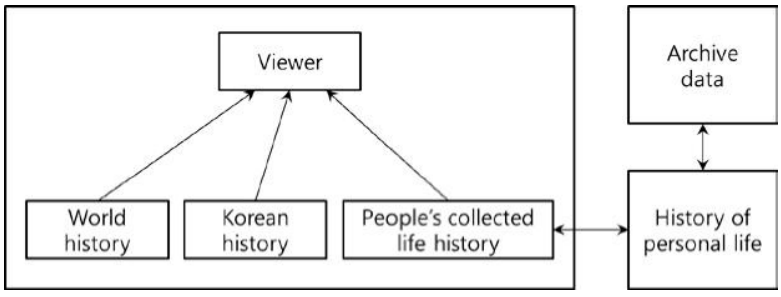

Figure 12. Architecture of Chronological Table

Figure 12 is architecture of chronological table. Viewer offers people's collected life history based on world history and Korean history. People's collected life history shows

the inserted Histories of personal life history in chronological order. It shows each History of personal life in accordance with archive data.

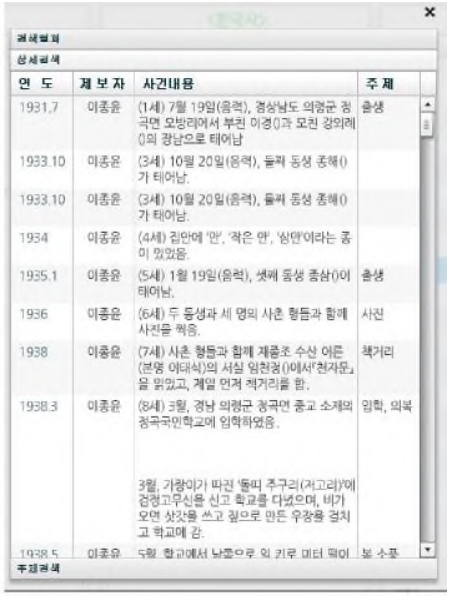

Figure 13. History of Personal Life

As illustrated in Figure12, History of personal life tells an informant's personal life from birth until the present in chronological order, showing how he has lived. For History of personal life, it primarily selected informants from the native-born who went through the Japanese colonial period and changing history. It allows an approach to their emotions and lives, not the general contents search. Instead of the simple search, the purpose of the system is to support more meaningful search and to show the flow of life by organizing the contents into the story.

3. 3. 3. Map Search based on GPS

Figure 14. Map Search based on GPS

In Figure 14, for Map search, the Raw Data Place from elements of extended Doublin Core was used. If there is GPS data, they are marked on the map using OpenAPI. In case GPS is absent, it converts its address to coordinates, using the address from the inserted data. To convert into coordinates, it sends the results in XML by calling Naver OpenAPI[12] and then Web protocol. It garners address by parsing the XML result.

The whole algorithm for generating Marker in the process is the following Pseudo-Code.

Load GPS and address data

FOR *i = 1 to N DO*

IF *(GPS_i == Null)* ***THEN***

Translate GPS_i to $address_i$ using Naver OpenAPI

GPS_i information stored in Database

ENDIF

$Marker_i$ representation using Map API

END FOR

After the data go through the algorithm, they are marked on a map as Marker, providing location information to users. The Server converts address data to GPS, which lets users know the location data where data were collected, using a map.

4. Implementation & Comparison

KTDAS is generated with the software platform like Table 4. For the server, it applied Linux (CentOs 6.3) that is the fully functional free operating system and provides high stability. Apache v.2.2.15 is used for WAS (Web Application Server). It used light and strong PHP v5.3.3 as a Web language. For the database, it used MySQL v.5.1.61 which is compatible with PHP and fully functional free RDBMS.

Table 4. Comparison of Search Results

Items	S/W
Operating system	Linux (CentOS 6.3)
WAS (Web Application Server)	Apache v.2.2.15
Web Language	PHP v5.3.3
Database	MySQL v.5.1.61

We compare KTDAS and other three systems in Table 5. All systems perform digitalizing the archive data that could be missed and lost in common. NDL deals with Archive materials of modern Japan. HOLOCAUST covers holocaust survivors. DENSHO takes imprisonment of American soldiers during WWII.

Table 5. Comparison of Search Results

Name	Subsection	Contents
KTDAS	Organization	Group for the People without history
	Web Site	http://minjung20.org/main/
	Subject	History of the ordinary people (life and labor, family and neighbors)
NDL	Organization	National Diet Library. Japan.
	Web Site	http://www.ndl.go.jp/modern/
	Subject	Archives material of modern Japan
HOLO CAUST	Organization	United States Holocaust Memorial Museum
	Web Site	www.ushmm.org
	Subject	Holocaust survivors
DENSHO	Organization	Densho
	Web Site	http://densho.org/
	Subject	Imprisonment of American soldiers during WWII

We compared the systems mentioned above, using items in Table 6. Table 6 has the item Archives and the item Search support. The item Archives is classified into types of data that each system provides, like Text, Oral, and Picture. Because most systems have similar features on this, they support most data types.

Table 6. Comparison of Search Results

Criterion		KTDAS	NDL	HOLO CAUST	DENSHO
Archives	Text	O	O	O	O
	Oral	O	X	O	O
	Picture	O	O	O	O
Search Support	General Search	O	X	O	O
	Detail Search	O	X	O	X
	Chronological Table	O	O	X	X
	Semantic Search	O	X	X	X
	Map Search	O	X	X	X

However, there are differences in the Search support. Search support is divided into Chronological table for time scale, Semantic search for the approach to semantics, and Map search based on location data. Most systems support simple search except that they partly provide some functions.

5. Conclusion

In this paper, we have introduced the Korean Traditional knowledge Digital Archives system (KTDAS) focusing on two respects: integrated structure and multi-functional application. We believe our archives system improves the preservation and application

of traditional knowledge. By doing so, human creativity and cultural diversity will be enhanced.

KTDAS suggested in this paper provides various types of search functions by digitalizing data about Korean peoples' lives in the twentieth century. KTDAS supports the various functions such as the simply search and the semantic search based on the query search of the Lucene search engine, and the detail search, the map search and the functions of the chronological table, based on that of the MYSQL database system.

The search methods that the system supports are rarely found in other systems. The approach brings a variety of search. Therefore, it suggests different interpretations from various angles with search of place, time, and semantics that have not been noticed in the existing system. However, the semantic search has a weakness that it is only for the text format. We will therefore study the semantic search for various formats, not to restrict to the text format in near future.

References

[1] H. H. Hahm and S. C. Park, "Digital Archives of Cultural Archetype Contents: Its Problems and Direction", In Journal of the Korean BIBLIA Society for library and Information Science, **(2006)**.
[2] NDL, National Diet Library. Japan, http://www.ndl.go.jp/modern/.
[3] HOLOCAUST, United States Holocaust Memorial Museum, http://www.ushmm.org/.
[4] Densho, http://densho.org/.
[5] A. D. Marwick, "Knowledge management technology", IBM System Journal, **(2001)**.
[6] S. Murugesan, "The Dublin Core: A Simple Content Description Model for Electronic Resources", Bulletin of the American Society for Information Science and Technology, **(2007)**.
[7] http://www.ad.jyu.fi/users/a/ankarjal/ITKD50/Dublin%20Core%20Metadata%20Template.htm.
[8] E. Hatcher and O. Gospodnetic, "Lucene in Action", Manning, USA, **(2004)**.
[9] L. C. Choi, K. U. Choi and S. C. Park, "An automatic semantic term-network construction system", International Symposium on computer Science and its Applications, **(2008)**.
[10] E. M. Voorhees, "Query expansion using lexical-semantic relations", Proceedings of the 17th annual international ACM, **(1994)**.
[11] P. Ogilvie, E. M. Voorhees and J. Callan, "On the number of terms used in automatic query expansion", Information Retrieval. Springer Press, **(2009)**.
[12] Naver Open API, http://dev.naver.com/openapi/.

Authors

Hoon Jo received his B.S degree division of Electronics and Information Engineering Department, Chonbuk National University, Jeonju, Jeollabuk-do, Korea, in 2012. His is a Master student in division of Electronics and Information Engineering Department, Chonbuk National University, Jeonju, Korea, from 2012 to current. His research interests include Information Retrieval, Data mining, Pattern recognition

Hong-chan Jeon received his B.S degree division of Electronics and Information Engineering Department, Chonbuk National University, Jeonju, Jeollabuk-do, Korea, in 2013. His is a Master student in division of Electronics and Information Engineering Department, Chonbuk National University, Jeonju, Korea, from 2013 to current. His research interests include Database, Information Retrieval, Web Search, Data structure and Security.

Han-hee Ham, cultural anthropology professor is now working in Chonbuk National University. Prof. Hahm is now involved in several research projects including Intangible Cultural Heritage Online Survey and Korea's Traditional Knowledge Archives since 2010. Her recent academic contribution is shown on the editorship of the book, The Understanding of Intangible Cultural Heirtage in Korea. Prof. Hahm earned her Ph.D. and M.A. in the Department of Anthropology, Columbia University. Prof. Hahm is currently Director of Research Institute of Intangible Cultural Heritage and Vice-president of Korean Society for Local History.

Soon-cheol Park received his B.S. degree in Applied Physics from Inha University, Incheon, Korea, in 1979. He received his Ph.D. in Computer Science from Louisiana State University, Baton Rouge, Louisiana, USA, in 1991. He was a senior researcher in the Division of Computer Research at Electronics & Telecommunications Research Institute, Korea, from 1991 to 1993. He is currently a Professor in the Division of Electronics and Information Engineering, Jeonbuk National University, Jeonju, Jeonbuk, Korea and Convergence Research Center (Information Telecommunications Research Center) member. His research interests include Pattern Recognition, Information Retrieval, Artificial Intelligence, Data Mining and Knowledge Discovery

