An Application of Text Mining: Bibliographic Navigator Powered by Extended Association Rules

Minoru Kawahara
Data Processing Center
Kyoto University
Kyoto 6068501, Japan
kawahara@kudpc.kyoto-u.ac.jp

Hiroyuki Kawano
Department of Systems Science
Kyoto University
Kyoto 6068501, Japan
kawano@i.kyoto-u.ac.jp

Abstract

In this paper, we discuss the implementation and performance of our developed bibliographic navigator with the text mining. We categorize the different attributes and extend the mining association algorithms and in order to provide more helpful rules. By our proposed algorithm, more interesting rules are derived from relationships between several categorized attributes in bibliographic databases such as INSPEC. We also evaluate the performance of our proposed algorithms on our developing bibliographic navigator and discuss the improvements of our navigator.

1. Introduction

Without the knowledge of stored data, it is generally difficult to retrieve appropriate results from databases. In the research fields of data mining[1, 3] and text mining [8], there have been many research efforts about effective algorithms to discover interesting rules in databases. Then we also extended the association rule [11], and we have been developing bibliographic navigator with the implementation of our proposed mining algorithms[6, 5].

Of course, there are many researches and implementations of intelligent search systems in order to dissolve or ease the difficulties of information retrieval[7, 10].

However, in this paper, we try to focus on characteristics of each attribute in bibliographic databases again, and we propose our mining algorithms in order to retrieve bibliographies effectively. Especially, our proposed algorithm derives association rules from not only single attribute but also relationship between several attributes, so query users can improve the given queries effectively by the rules.

Furthermore, we implemented our proposed algorithms in a practical bibliographic navigator using Opentext which is a famous and conventional full text search system. We also evaluate the performance of our developed system using INSPEC database.

In Section 2, we propose a weighted mining association algorithm. In Section 3, we propose algorithms which expands keyword space by using relationship between several attributes. In Section 4, we show the structure of our practical bibliographic navigator. In Section 5, we evaluate the performance of our practical system. In Section 6, we discuss to improve our system in a view point of materialization. Finally, we make concluding remarks in Section 7.

2. Weighted mining association algorithm

In this section, we extend the mining association algorithm [11], in which there have been many research efforts, considering weight of each keyword in a database.

The mining association algorithm requires two values $support$ and $confidence$ to derive rules. Given a set of transactions, where each transaction is a set of items, an association rule is an expression $X \Rightarrow Y$, where X and Y are sets of items. For example, keyword becomes item. The intuitive meaning of such a rule is that transactions in the database which contain the items in X tend to also contain the items in Y. The $support$ of the rule $X \Rightarrow Y$ is the percentage of transactions that contain both X and Y, hence $P(X \cap Y)$. And the $confidence$ of the rule $X \Rightarrow Y$ is the percentage of transactions that contain Y in transactions which contain X, hence $P(Y \mid X)$.

However in huge scale databases, the $support$ tends to become too small to derive rules by the original algorithm. So we redefine the $support$ of the rule $X \Rightarrow Y$ as the ratio of $WEIGHT(X \cap Y)$ to $MAX(X)$, where $WEIGHT(X \cap Y)$ is the total of the minimum weight in transactions that contain both X and Y, and $MAX(X)$ is the total of the maximum weight in transactions that contain any items in X. The intuitive meaning of such the $support$ is that transactions in the database which contain the items

in both X and Y have a weight $WEIGHT(X \cap Y)$ in transactions concerning X.

Adapting these algorithms on bibliographic database, we adjust keyword to item and adjust tuple to transaction in the database. Let's assume the following:

- $\mathcal{G}$: a set of keywords given in a query.
- $\mathcal{O}$: a set of the keyword sets derived from $\mathcal{G}$ as association rules in the database.
- $\mathcal{T}_g$: a set of tuples which contain $\mathcal{G}$ in the database.
- $\mathcal{T}_a$: a set of tuples which contain any keywords in $\mathcal{G}$ in the database.
- $\mathcal{K}_a$: a set of all keywords in $\mathcal{T}_a$.
- $\mathcal{K}_c$: any combinations of $\mathcal{K}_a$.
- $\mathcal{T}_c$: a set of tuples which contain both $\mathcal{G}$ and $\mathcal{K}_c$.
- $\mathcal{U}$: the set of all tuples in the database.

$Support$ of $\mathcal{K}_c$, which is denoted by $support(\mathcal{K}_c)$, of the mining association algorithm is given by the ratio of $\mid \mathcal{T}_c \mid$ to $\mid \mathcal{U} \mid$:

$$support(\mathcal{K}_c) = \frac{\mid \mathcal{T}_c \mid}{\mid \mathcal{U} \mid}. \quad (1)$$

However $\mathcal{U}$ is huge in a huge scale database and $support(\mathcal{K}_c)$ tends to become too small by the original algorithm. We improve this problem and consider weight of keyword to extend the original algorithm, and we call our proposed algorithm as "Weighted Mining Association Algorithm". In order to prevent $support$ from being too small, the retrieval space is reduced into $\mathcal{T}_a$ which concerns any keywords in $\mathcal{G}$ in the database then the denominator of the equation (1) becomes smaller. Moreover, we define that if a keyword $k_j (i \in J)$ appears w_{ij} times in a tuple $T_i (i \in I)$ then the keyword k_j has the weight w_{ij} in the tuple T_i, where I is a constant of tuples and J is a constant of keywords in a tuple. Keyword-weight combination (k_j, w_{ij}) may be given as some weight of keyword. Therefore $support$ of $\mathcal{K}_c$ can be given by:

$$support(\mathcal{K}_c) = \frac{W(\mathcal{K}_c)}{W_a}, \quad (2)$$

$$W(\mathcal{K}_c) = \sum_{\mathcal{T}_c} \min_{\mathcal{K}_c} w_{ij},$$

$$W_a = \sum_{\mathcal{T}_a} \max_{\mathcal{K}_a} w_{ij}.$$

$Confidence$ of $\mathcal{K}_c$ of the weighted mining association algorithm is the same as the mining association algorithm and is given by:

$$confidence(\mathcal{K}_c) = \frac{\mid \mathcal{T}_c \mid}{\mid \mathcal{T}_g \mid}. \quad (3)$$

Thus if $\mathcal{K}_c$ has both $support(\mathcal{K}_c)$ not less than $Minsup$ and $confidence(\mathcal{K}_c)$ not less than $Minconf$, then the keyword set $\mathcal{K}_c$ is stored into $\mathcal{O}$ as an association rule enough to cause demands. Where $Minsup$ is a minimum threshold of $support$ to judge whether the keyword set holds enough retrieval needs, and $Minconf$ is a minimum threshold of $confidence$ to judge whether the keyword set holds enough confidence.

Table 1. Categorization of attribute types.

Category	Meaning
Characteristic	This includes keywords which show the characteristics of a bibliography. These values are provided by the authors, the publishers and the database editors. ex. Title, Keyword
Description	This includes sentences, phrases and words which describe the contents of a bibliography. ex. Abstract, Table of contents, Index.
Supplement	This shows the supplement information of a bibliography. ex. Author, Publisher, Conference, ISBN

3. Association rules using several attributes

Basically, we implemented the weighted mining association algorithm in our bibliographic navigator constructed in this paper. In addition to the algorithm, we categorize attributes in bibliographic database by their semantic properties and use the relationships between several categorized attributes in order to derive effective rules in our system. It is also processed to expand keyword spaces using the relationship between several categorized attributes[5]. Now we define keyword space as domain which includes keywords concerning retrievals, such as query and attribute.

3.1. Categorization of attribute types

In order to expand keyword spaces appropriately and use several attributes effectively, we categorize attributes in bibliographic database into three categories shown in Table 1 by their semantic properties. And we use characteristic and descriptive types of attributes to derive meaningful association rules by join operations in these types.

However characteristic type of attributes in bibliographic database, such as Title and Keyword, do not have enough number of keywords to derive association rules although these attributes contain keywords which are strongly associated with a bibliography. That is, the keyword space of

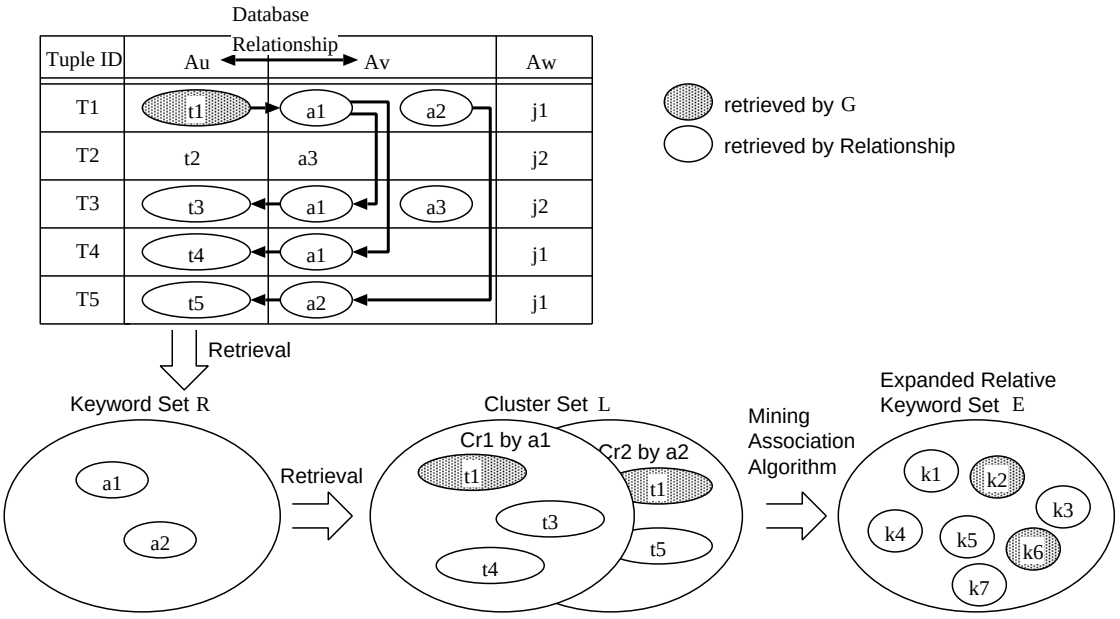

Figure 1. Expansion of keyword spaces using the relationship between attributes.

characteristic type of attributes is small as shown in Table 2. Thus it is also required to expand a keyword space using an attribute related to the attribute [5].

Moreover, mining association algorithms tend to derive meaningless keywords such as "of", "the" and "and", so it is required to avoid to derive such ineffective keywords. Hence in this paper, we use dictionaries containing meaningless keywords to remove such ineffective keywords, and apply the weighted mining association algorithm on several categorized attributes.

Table 2. Keyword spaces of the categorized attributes in INSPEC database.

Category	Keyword space	Average
Characteristic	Small	Title 11
		Keyword 30
Description	Small $\sim$ Large	Abstract 183
Supplement	Small	Author 3

3.2. Expansion of keyword spaces

As shown in Table 2, characteristic and supplement types of attributes generally do not have enough number of keywords for mining association algorithms, for example the attribute Title has 11 keywords on average and the attribute Author has only 3 keywords, which are authors, on average. That is, keyword spaces of them are so small that it is hard to derive effective keywords from such attributes although they are strongly associated with bibliographies.

Therefore we propose an algorithm to expand keyword spaces by using the relationship between these types of attributes, for example A_u and A_v shown in Figure 1. In Figure 1, it is assumed that a keyword set $\mathcal{G}$ is given in a query and t_1 which is a value of A_u in a tuple T_1 contains $\mathcal{G}$. If a relationship between A_u and A_v is given to the database, then a_1 and a_2 which are values of A_v in the tuple T_1 are gathered into a keyword set $\mathcal{R}$ like as $\{a_1, a_2\}$ by the relationship. Next, tuples which contain each keyword in $\mathcal{R}$ in the attribute A_v are retrieved and the values of A_u are gathered into a cluster, for example $\{t_1, t_3, t_4\}$ from a_1 and $\{t_1, t_5\}$ from a_2. And the clusters are stored into a cluster set $\mathcal{L}$ as the following process:

$$t_1 \rightarrow \{a_1, a_2\} \rightarrow \{\{t_1, t_3, t_4\}, \{t_1, t_5\}\}.$$

Thus treating the cluster as a domain, the weighted mining association algorithm is applied on the cluster set $\mathcal{L}$ to derive rules $\mathcal{E}$.

Attaching the attribute Title to A_u and the attribute Author to A_v, it is possible to expand the keyword space of Title. This means to construct a keyword space of bibliographies written by an author. Generally, query users for a bibliographic database are often interested in bibliographies written by an author, thus it is an appropriate way to give a relationship between attributes such as Title and Author. Our proposed algorithm can be summarized in the following:

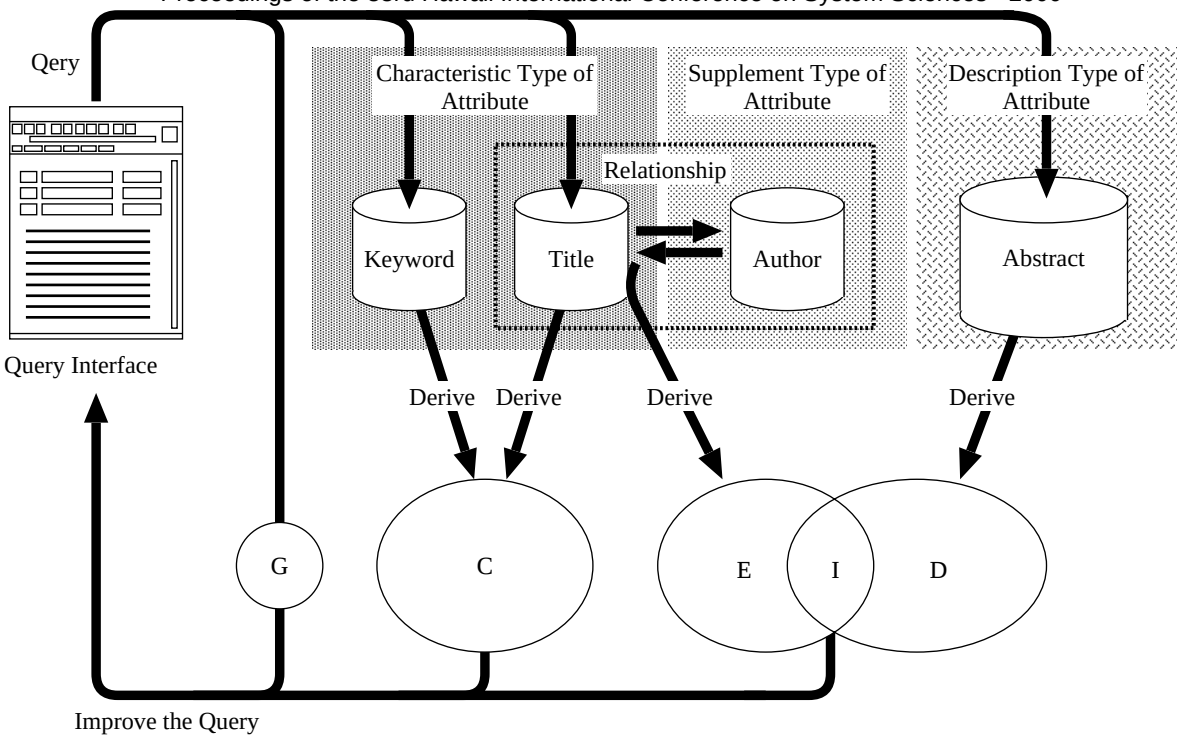

Figure 2. Improvement of queries using the relationship between several attributes.

Algorithm 1

Input:

A keyword set $\mathcal{G}$ given in the input query,
A characteristic type of attribute A_c,
A characteristic type of attribute A_u, and a attribute A_v related to A_u.

Output:

An expanded keyword set $\mathcal{E}$

Method:

1. Select tuples which contain $\mathcal{G}$ in A_u and construct a tuple set T_g.
2. Gather the keywords from A_v in T_g and construct a keyword set $\mathcal{R}$.
3. Select tuples which contain a keyword in $\mathcal{R}$ and construct a tuple set T_r.
4. Gather the keywords from A_u in T_r and construct a cluster C_r.
5. Repeat Step 3 and Step 4 for each keyword in $\mathcal{R}$ and store the clusters into a cluster set $\mathcal{L}$.
6. Apply our mining association algorithm on $\mathcal{L}$ and derive a keyword set $\mathcal{E}$.

3.3. Relationship between several attributes

Characteristic type of attribute contains the value given by the bibliography author and the database editor, hence it is thought that the attribute gives a keyword set $\mathcal{C}$ which contains keywords strongly associated with the bibliography. Next, applying the algorithm 1 in a database, an expanded relative keyword set $\mathcal{E}$ is derived. The keyword set $\mathcal{E}$ is derived from a relationship based on characteristic type of attribute, hence it is thought that $\mathcal{E}$ contains effective keywords.

However the keyword spaces of $\mathcal{C}$ and $\mathcal{E}$ are still small because the attributes do not have enough keywords. In order to expand the keyword space, we use a description type of attribute to derive a relative keyword set $\mathcal{D}$. The keyword set $\mathcal{D}$ derived by this operation dose not always contain rules which are relative to the subject of bibliographies. Thus it is thought that $\mathcal{D}$ contains more common rules than $\mathcal{C}$ and $\mathcal{E}$. But we have to give attention to that common rules are not always next to keyword spaces in which query users are interested, for example an attribute such as Abstract often includes negative keywords to show advantages of the document.

Considering the semantic properties of attributes as shown in Figure 2, we propose an algorithm which derives a relative keyword set effectively in the following:

Algorithm 2

Input:

A keyword set $\mathcal{G}$ given in the input query,
A characteristic type of attribute A_c,
A characteristic type of attribute A_u, and a attribute A_v related to A_u,
A description type of attribute A_d.

Output:

An relative keyword set $\mathcal{O}$

Method:

1. Derive a keyword set $\mathcal{C}$ from A_c.
2. Derive a expanded keyword set $\mathcal{E}$ from the relationship between A_u and A_v applying the algorithm 1.
3. Derive a keyword set $\mathcal{D}$ from A_d.
4. If the intersection $\mathcal{I}$ of $\mathcal{E}$ and $\mathcal{D}$ produces an empty set, then the minimum $support$ threshold $Minsup$ and the minimum $confidence$ threshold $Minconf$ are lowered.
5. Go to step 1 if $Minsup$ and $Minconf$ are not lower than the limits given by the system administrator.
6. Calculate $\mathcal{O} = \mathcal{C} \cup \mathcal{I}$.
   Output the relative keyword set $\mathcal{O}$.

In the step 4, $Minsup$ and $Minconf$ are changed dynamically according to the derived association rules. The reason for using this method is that there is anxiety that a low threshold tends to derive many relative keywords and effective keywords may be buried under them and be overlooked, and a high threshold tends to remove even effective keywords. Setting lower the threshold gradually according to the common association rules in this way, it is possible to minimize effective keywords to be removed and maximize effective keywords to be derived. And it is possible to give queries using appropriate derived keywords.

4. Structure of bibliographic navigator

For our practical bibliographic navigator, we selected INSPEC database as a bibliographic database. Our system handles 2,085,629 titles which have been published from January 1990 to May 1997.

Our system consists of three modules: (a) a query interface system, (b) a full text search system and (c) a mining association system as shown in Figure 3. Hence two databases: a conventional full text search bibliographic database and a mining association database. Both of the

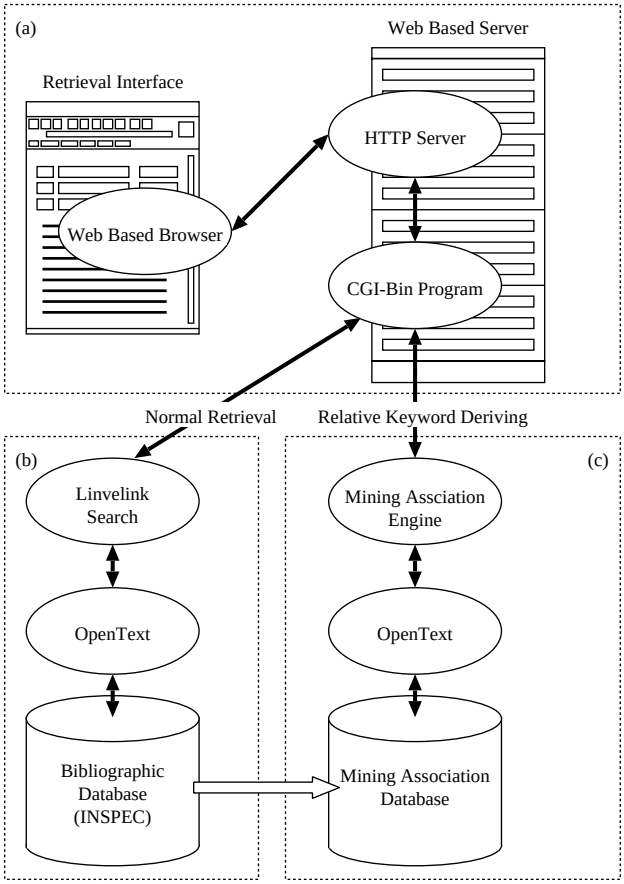

Figure 3. The structure of bibliographic navigator

database systems use OpenText, which is developed as a full text search system in Canada, as the database management system (DBMS).

Our system executes the follows in a retrieval:

(a) Query interface system: Using a user interface as shown in Figure 4, the CGI-Bin program receives a query from a Web browser and analyzes the query, and sends the analyzed query to the full text search system and the mining association system in parallel. When the CGI-Bin receives results from both systems, it constructs a HTML formed document from the results and returns it to the Web browser. On the Web browser, our system gives knowledge to query users by displaying the relative keywords, by which the original query can be improved.

(b) Full text search system: A full text search on INSPEC data is executed by OpenText. OpenText receives a query through Livelink Search which is the HTTP in-

Table 3. A example of a translation between SGML and HTML

SGML	HTML
<Title>	<HR><FONT COLOR="#002255"> Title : </FONT> %s
<Author>	<FONT COLOR="#002255"> Author : </FONT> %s
<Abstract>	<FONT COLOR="#002255"> Abstract : </FONT> %s
<Publisher>	<FONT COLOR="#002255"> Publisher : </FONT> %s
<ConferenceDates>	<FONT COLOR="#002255"> Conference Dates : </FONT> %s

Figure 4. The query window of our system.

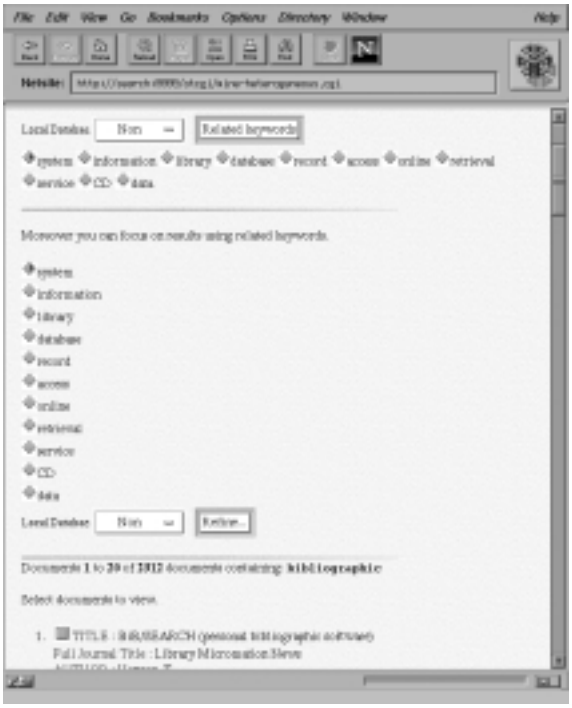

Figure 5. The result window of our system.

terface for OpenText. And OpenText returns the result for the query to the CGI-Bin program.

(c) Mining association system: Our mining association algorithms shown in Section 2 and Section 3 are executed by the mining association engine. The engine derives a relative keyword set and returns it to the CGI-Bin Program.

4.1. Bibliographic database

Our bibliographic database contains 2,085,629 bibliographic titles even though a part of all INSPEC data is used, and the volume of data is 6GB and the volume of index is 3.5GB. Hence it takes much time to execute a full text search, so we are using OpenText because OpenText is one of the highest performance full text search systems on a huge database and can also handle SGML formed data.

As the data form in our database, we selected SGML(Standard Generalized Markup Language), which was defined in ISO standard 8879:1986, for portability corresponding to many kind of retrievals. As shown in Figure 6, a SGML document consists of three items: a SGML Declaration which describes the rules of the current document such as a character set and characters used as control characters, a DTD (Document Type Definition) which defines the structure of the current document such as tags and relationships between the entities, and a document instance which is written in text.

SGML is also familiar with Web browsers used as the query interface. Because HTML (Hyper Text Markup Language), which is used as a Web document form, is one variation of SGML adjusted to Web data interchange by a DTD, thus it is easy to make a translation between the SGML form and the HTML form, for example in Table 3 the left hand

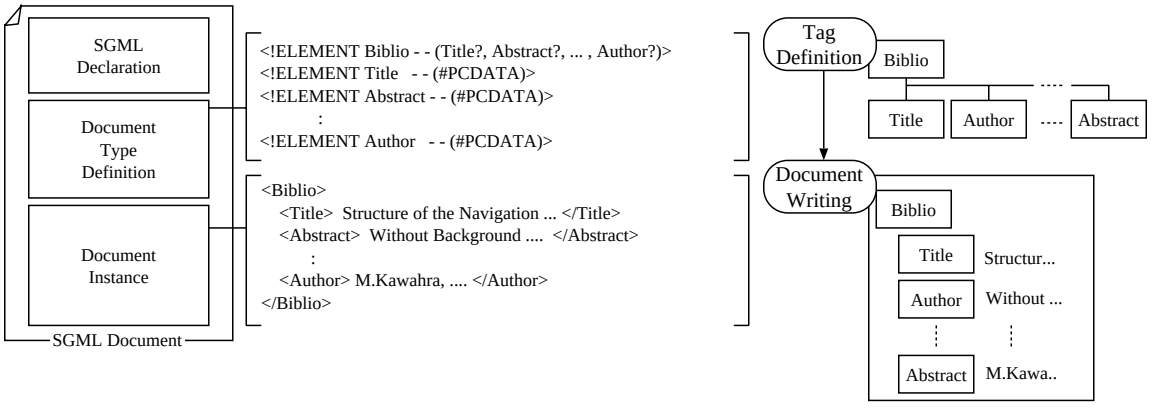

Figure 6. The data form of our bibliographic database

side is the SGML form and the right hand side is the HTML form.

As a SGML Declaration, we basically use the basic SGML documents shown in ISO standard 8879 and modified a part of it adjusting to our system. We changed value of the maximum number "ATTCNT" of tokens from 40 to 200 and changed value of the maximum length "NAMELEN" of token name from 8 to 100 in "SYNTAX".

As a DTD, we defined items corresponding to all tags used in INSPEC data for example as the following:

```
<!ELEMENT INSPECData - - (Biblio*)>
<!ELEMENT Biblio - - (AccessionNumber?,
AmendmentDate?, RecordType?,
CopyrightStatement?, Title?, Abstract?,
                  ⋮
<!--  ELEMENTS MIN CONTENT(EXCEPTIONS) -->
<!ELEMENT AccessionNumber - - (#PCDATA) >
<!ELEMENT AmendmentDate   - - (#PCDATA) >
<!ELEMENT RecordType      - - (#PCDATA) >
                  ⋮
```

In the first line a declaration of a tag enclosing document instances is appeared for example `Biblio*`. And in the second line a declaration of tags contained in a document instance, which is a bibliographic data, for example `AccessionNumber`. In the lines following them there are declarations corresponding to each tags in INSPEC data.

According to the DTD, INSPEC data was converted in the SGML form and stored into files. The following is a example of a file which contains document instances:

```
<Biblio>
<AccessionNumber>5512630</AccessionNumber>
<RecordType>02</RecordType>
<CopyrightStatement>Copyright 1997, IEE
</CopyrightStatement>
<Title>Data mining with composite events
 based sampling in a dynamic environment
</Title>
<Author>Kawano, H. Hasegawa, T.</Author>
<Abstract>Data mining, or knowledge
 discovery in databases, is the
                  ⋮
</Biblio>
```

Table 4. Correspondence of attributes to keyword sets

Attribute	Tag	Keyword set
Title	Title	$\mathcal{C}_1$
Keyword	Keyword	$\mathcal{C}_2$
Author (as a related attribute)	Title $\Rightarrow$ Author $\Rightarrow$ Title	$\mathcal{E}$
Abstract	Abstract	$\mathcal{D}$

4.2. Mining association database

We selected attributes shown in Table 4 for the mining association system. And we gave a relationship between Title and Author for expanding keyword spaces of the attribute Title by the algorithm 1 because query users are often interested in bibliographies written by an author. Next, we retrieved the attributes from the bibliographic database to count each keyword appeared in an attribute as the weight in the attribute, and stored the keyword-weight combinations $(keyword, weight)$ into the mining association database.

As the data form in the mining association database, we selected the SGML form the same as the bibliographic database. This enables that a kind of DBMS is commonly used in both database, and OpenText was selected. And we are also using the same SGML Declaration and DTD as the bibliographic database. The volume of data became 3GB

and the volume of index became 1.4GB. The following is a example of a document instance in the mining association database:

```
<!DOCTYPE INSPECData SYSTEM "inspec.dtd">
<INSPECData>
<Biblio>
                    ⋮
<Biblio>
<AccessionNumber>5512630</AccessionNumber>
<Title>a:1 based:1 composite:1 data:1
 dynamic:1 environment:1 events:1 in:1
 mining:1 sampling:1 with:1  </Title>
<Author>Kawano, H. Hasegawa, T.</Author>
<Abstract>a:3 adopt:1 algorithm:1 and:2
 anomaly:1 applications:1 area:1 as:1
 attribute:1 based:2 be:1 business:1
                    ⋮
</Biblio>
</INSPECData>
```

Using this database, the mining association engine derives a relative keyword set applying the proposed algorithms in Section 2 and Section 3. The mining association engine consists of a retrieval part of the keyword-weight combinations $(keyword, weight)$ and a mining association part. The programs are written in programming language C and shell script.

Meaningless keywords in the keywords which are retrieved as values in Title, Keyword and Abstract in Table 4 are removed by using dictionaries. And then the weighted mining association algorithm is applied on these attributes and keyword sets $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{D}$ are derived respectively. But each number of sampling tuples retrieved in the database is limited not to exceed 2,048 for response time.

On the other hand, the relationship between Title and Author is used for expanding the keyword space of Title by the algorithm 1. Authors is derived by the mining assciation algorithm from the keyword (author) set retirieved by the relationship. But the number of authors used for the algorithm 1 is limited to maximum appeared 20 persons because there are too many retrievals for all authors to derive association rules in limited time. It is also done that meaningless keywords are removed using dictionaries before applying the algorithm. As a result, a keyword set $\mathcal{E}$ is derived.

Finally, the algorithm 2 is applied on the results to derive a relative keyword set $\mathcal{O} = (\mathcal{C}_1 \cup \mathcal{C}_2) \cup (\mathcal{E} \cap \mathcal{D})$.

4.3. Web based interface

As the user interface for our system, we selected the Web interface using the HTML form so that operations for our system do not depend on any platforms. As shown in Figure 4, we made the look and feel of the query window to be the same as the conventional full text search bibliographic system.

Table 6. Derived keyword sets from keyword "bibliographic"

Keyword set	Keywords
$\mathcal{O}$	system, information, library, database, record, access, online, retrieval, service, CD, data
$\mathcal{C}_1$	database, information, system, library, record, online, retrieval, service
$\mathcal{C}_2$	database, information, library, system, online, record, retrieval, service, CD, data
$\mathcal{E}$	retrieval, system, hypertext, based, information, library, structure, linkage, semantics, searcher, rural, response, resources, performance, model, issue, effect, education, design, access, united, $\cdots$
$\mathcal{D}$	library, information, database, system, record, data, access, CD

A retrieval is started by clicking the button `Search` on the query window with filling the keyword input fields by a user. So a result is displayed on the result window as shown in Figure 5. On the result window, the keywords which are relative to the keywords in the input query are displayed, then the user can improve the query by clicking the button `Refine...` and remake the query by clicking the button `Related keywords`.

Now, a part of a bibliographic title is shown in Figure 5 and it is possible to show the detail of that. There is also a function to show titles which are required by a query user in a lump. It is possible to use almost functions of the query interface by operation of a mouse.

5. Performance evaluation

For example, a result of a query with "bibliographic" is shown in Figure 5. The result window provides relative keywords as $\mathcal{O}$ in Table 6 derived from "bibliographic" by our system. Each derived keyword set is also shown in Table 6 and it is found that $\mathcal{E}$ contains many different keywords from the other sets: $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{D}$.

Furthermore, in order to evaluate the performance of our system, we analyzed the average computing cost to derive rules from commonly used 1,000 keywords in Tile of INSPEC database. The reason why we selected the commonly used 1,000 keywords for the evaluation is that the keywords used in INSPEC database are biased in particular words as shown in Figure 7, which is sorted in order of frequency.

Our hardware specification for the evaluation is the

Table 5. Performance of computing cost for our system.

Order	Normal [sec]	Total [sec]	Total/Normal [times]	Algorithm 1 [sec]	Algorithm 1 [%]	Algorithm 2 [sec]	Algorithm 2 [%]
1 - 100	27.8	54.7	2.0	17.6	27.9	10.4	20.3
101 - 200	27.5	56.5	2.1	17.7	28.0	10.9	20.1
201 - 300	27.3	58.5	2.1	19.4	29.7	10.8	19.2
301 - 400	26.8	63.5	2.4	25.1	34.6	10.3	17.6
401 - 500	26.6	66.6	2.5	27.0	36.1	10.5	17.0
501 - 600	27.2	65.3	2.4	24.2	32.3	10.9	18.1
601 - 700	26.6	69.2	2.6	28.4	36.0	10.6	16.6
701 - 800	27.0	67.8	2.5	26.6	34.2	10.6	17.0
801 - 900	26.8	69.1	2.6	28.0	35.9	10.6	16.7
901 - 1000	26.4	71.4	2.7	31.2	38.8	10.7	16.2

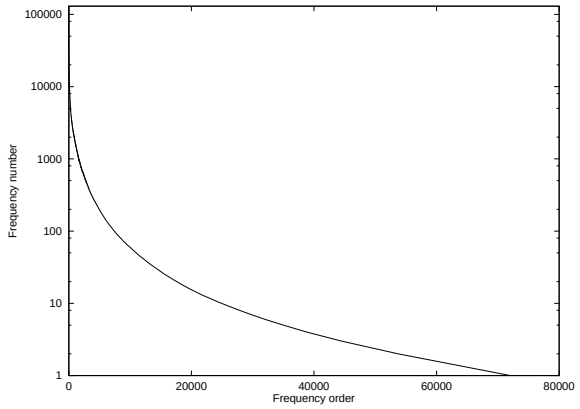

Figure 7. The frequency of words appeared in titles.

following: Sun Microsystems Ultra-2/1300 (300MHz UltraSPARC-II, 128MB memory) as a CPU and Ciprico Rimfire 6710 (Fast/Wide SCSI differential, RAID3, 32GB) as a disk system. The result is shown in Table 5. In Table 5, "Normal" means the time for the full text search only, "Total" means the time for our system, and "Total/Normal" shows the ratio of "Total" to "Normal". Moreover, "Algorithm 1" is the time for executing the algorithm 1, and "Algorithm 2" is the time for executing the algorithm 2 without the cost of the algorithm 1 in our system.

Looking at the ratio "Total/Normal" in Table 5, it is found that the cost of our system is only from 2 to 3 times as much as the full text search system. Thinking about accessing to the database for four attributes and a related attribute, the cost can be kept low. Comparing the cost of the algorithm 1 with the cost of "Total", it is also found that the former shares almost the increased cost in the latter but the cost by the algorithm 2 is constant. It is thought that the lower frequency is, the fewer tuples containing the keyword are, hence the $support$ of a retrieved author is higher and it is increased authors who are required to be retrieved, and this leads to increase access times to the database.

6. Discussion

Although the cost of our system is only from 2 to 3 times as much as the full text search system as shown in Table 5, it forces query users to wait so long time that they can't bear, for example about 60 seconds. Thus it is comparatively expensive to derive association rules and it is required to reduce this deriving cost.

It has been frequently discussed in the research field of data mining about this kind of deriving cost and there have been many research efforts about sampling method in even the first stage. Sampling is a method which reduces the deriving cost to pick up appropriate numbers of items from whole data and process the items for data mining. It has been studied widely about caching for speeding up query processing [9]. Caching is a method which stores the query results to a cache space and return the result from it if the same query previously executed. So we have tried to implement several kinds of methods, but there are some problems of them, for example sampling doesn't always derive exactly rules because of the result items of the sampling, and it is required to clear the cache space when database is updated.

Recently, there has also been research efforts about parallel processing for speeding up driving processing using parallel computers and cluster computers [2]. It seems that the database access is a bottleneck since our algorithm causes more than eight database accesses and more than five times as many processes as the number of accesses in parallel. Thus we think that parallel processing will be a solution for the bottleneck.

By the way, there has been research efforts about effective pre-processing, which is called as materialization, and updating method on huge data from a point of data cube view [4] because in a data warehouse updates come in bulk. So it is thought that materialization will be used as one of

methods which reduce the deriving cost. If our system materializes in advance the association rules from keywords which will be used in queries and scans the materialized association rules to return the result in a deriving, the waiting time of users will be reduced dramatically.

7. Conclusion

The recent permeation of computer environment leads that digital libraries and desk top publishing are payed attention. However it is still difficult to provide the retrieval system that makes query users to retrieve appropriate bibliographic data effectively. It is also important to show a guideline to construct such a effective system that is a solution for the current problems because it is increasing retrievals by free words not to analyze the subject of bibliographies and not to unify the keywords of bibliographies.

In this paper, we adopted the data mining technologies, which would be basic solutions for huge data processing systems, and tried to construct a bibliographic navigator, in which query users could retrieve bibliographies effectively even without background of retrieval systems and domain knowledge. Furthermore, it was found that the deriving cost of our proposed weighted association rules was kept low and it would be possible to execute the algorithm in limited time on current computer systems. It was also possible to derive association rules effectively by expanding keyword spaces using the relationship between attributes. On these methods, we actually constructed a bibliographic navigator and could show that query users could get improved queries easily and good results by our system. And query users could have a chance to know vacillation of the keywords in the query.

At present, we are evaluating a materializing algorithm for our mining association algorithm to reduce the derived time for users. In the future, we also need to clear the advantages of other algorithms, such as parallel algorithm and clustering algorithm, for navigator and make our current system to be more effective navigator system

Acknowledgment

A part of this work is supported by the grant of Scientific Research (10780259, 08244103) from the Ministry of Education, Science, Sports and Culture of Japan. We are grateful to Nissho Iwai Infocom Co., Ltd. for the source programs of the full text search system “OpenText”.

References

[1] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1996.

[2] A. A. Freitas. Scalable, high-performance data mining with parallel processing. *Second Symposium on Principles of Data Mining and Knwoledge Discovery (PKDD-97)*, 1998.

[3] J. Han, S. Nishio, H. Kawano, and W. Wei. Generalization-based data mining in object-oriented databases using an object cube model. *Data and Knowledge Engineering*, (25):55–97, 1998.

[4] V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing data cubes efficiently. *Proc. of the 1996 ACM SIGMOD*, pages 205–216, 1996.

[5] M. Kawahara, H. Kawano, and T. Hasegawa. Data mining technologies for bibliographic navigation system. *Transactions of the IPSJ*, 39(4):878–887, 1998.

[6] H. Kawano. Mondou: Web search engine with textual data mining. *Proc. of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pages 402–405, 1997.

[7] K. Parsaye, M. Chignell, S. Khoshafian, and H. Wong. *Intelligent Databases*. John Wiley & Sons, Inc., 1992.

[8] R.Feldman. Practical text mining. *Second Symposium on Principles of Data Mining and Knwoledge Discovery (PKDD-97)*, 1998.

[9] N. Roussopoulos. Materialized views and data warehouses. *SIGMOD Record*, pages 21–26, 1998.

[10] G. Salton. Another look at automatic text-retrieval system. *Communications of the ACM*, 29:648–656, 1987.

[11] R. Srikant and R. Agrawal. Mining generalized association rules. *Proceedings of the 21st VLDB*, pages 407–419, 1995.