1998 International Computer Symposium
Workshop on Software Engineering and Database Systems
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

A COLLABORATIVE INTERNET DOCUMENTS ACCESS SCHEME USING ACIRD

Shian-Hua Lin, Chi-Sheng Shih, Meng Chang Chen*, Jan-Ming Ho*, Ming-Ta Ko*, and Yueh-Ming Huang*
Department of Engineering Science, National Cheng Kung University, Tainan, Taiwan
Institute of Information Science, Academia Sinica, Taipei, Taiwan*

ABSTRACT

In this paper, we present a collaborative intelligent Internet multi-web sites documents search system using ACIRD. ACIRD is a system that automatically learns the classification knowledge from Web pages and applies the knowledge to automatic classification of Web pages to some classes in a class hierarchy. Data mining technique is used to learn the association of terms to discover the hidden semantic connections between terms. With the capabilities of ACIRD, it is straightforward to extend ACIRD to collaborate multi-web site document access. Based on the learned classification knowledge, a collaborative two-phase search engine is proposed, which dispatches queries to distributed Web sites to match documents and presents hierarchically navigable results to the Internet users rather than conventional ranked flat results.

1. INTRODUCTION

The rapid growth of the Internet has changed the way of working and living that the Internet becomes a major source of information and means of communication. However, the excessive information on the Internet creates the information overflow problem. As a result, information retrieval (IR) systems (or called search engines) come to help the Internet users to alleviate the problem. The conventional IR systems are designed to facilitate rapid retrieval of information for diverse users. By applying word-based index/search mechanisms [7], words in a document are extracted, indexed and stored in databases. The indexes are later employed in retrieving documents relevant to a query represented by terms.

From the perspective of retrieval effectiveness and efficiency, word-based IR systems are proficient in accessing a large document base. However, for a query with two words[1] submitted to a word-based search engine implemented with vector space model (VSM) [11,12], more than thousands of documents are probably retrieved. Ranking a large number of documents using very few keywords may not produce an ordered sequence of documents meeting the preference of the user. Consequently, user has to retrieve many irrelevant documents before obtaining the desired information. Also, the conception gap between Web developers and the Internet users enlarges the difference between the retrieved results and user's expectation. Therefore, many Internet word-based search engines usually retrieve thousands of documents with few desired, while desired documents may not be retrieved. For instance, the term "airline schedules" in documents does not fully match the term "flight schedules" in query, but both terms represent the same semantics. In a specific IR environment, a thesaurus database with terms for the special domain knowledge can alleviate such problem. However, because of the diverse contexts of the Internet, no static thesaurus can cover the mismatching and shifting semantics of terms. Therefore, the Internet IR systems should be able to search relevant documents efficiently, rank and organize the documents in accordance with user's expectation.

As for the coverage of the Internet search engines, the reality is that a single search engine (or called *server*) can not handle a very large number of queries without a powerful server or a group of servers. Furthermore, empirical results show that no single search engine is likely to return more than 45% of the relevant result [13] suggesting that simultaneous query of several independent servers may return a more satisfactory result. The autonomy of web sites, which is along the line of the border of organization, demands a mechanism to collaborate among the Web servers. To solve the problems, many researchers have advocated the deployment of special search engines based on geographic locations to distribute the load and to construct an integrated gateway to these databases. Meta-search engines such as All-In-One [5], META Search [15], and MetaCrawler [13,14] can reduce the burden of the user that make available search engines that may have been unknown to the users. The search engines handle the simultaneous submission of queries; some direct the query to appropriate engines and some post-process the results as well via a single interface [3]. Unfortunately, uncontrolled meta-search can lead to the "tragedy of the commons" problem from economics in which an individual's best interests counter to society's that individual users appear to be served by simultaneously searching every possible search engine on the Internet for desired information.

A meta-search system can be a good Web citizen [6] by querying only those search engines likely to return useful results. We consider a system composed of sites running ACIRD search engines (called *ACIRD components*), and a central site (called *ACIRD central site*) running a meta-level control mechanism. The ACIRD component acts almost the same as a stand-alone ACIRD system that learns classification knowledge from Web documents and forwards the knowledge to the central site. The ACIRD central site collects the classification knowledge from the remote ACIRD engines and integrates the knowledge into

[1] According to the statistics in [4], the average query length is 1.3 words.

a global view. When processing a user query, the ACIRD central size decides the ACIRD engines that may return useful information, and prepares a separate query for each of them. Then it collects the returned result and summaries the return to the users. The retrieval process also allows user interaction, so called *two-phase search*, to select the interesting web site for further exploration. With such archiecture, the system allows both the load sharing and collaborative access from a set of web sites.

In the rest of the paper, the ACIRD engine is reviewed in Section 2. In order to illustrate the system clearly, in Section 3, we describe the learning model of ACIRD. In Section 4, the experiments of automatic classification to justify the design of ACIRD. Then the collaborative two-phase search approach is introduced in Section 5. Finally, we conclude the contribution of the paper.

2. THE ACIRD SYSTEM

ACIRD is designed to automatically classify documents. It is motivated to improve the performance of the current manual classifications at Yam. The classes lattice of Yam with generalized/specialized operators is called *ACIRD Lattice* in the paper. The classification process of ACIRD is composed of two phases: *training phase* and *testing phase*. In the first phase, documents with their manually assigned classes in Yam are employed as the training set to learn the classification knowledge of the classes in the lattice. Then the newly collected documents manually categorized by human experts are applied to verify the learned classification knowledge in the second phase. Based on the classification knowledge, *ACIRD Classifier* automatically classifies the Web documents to proper classes in ACIRD Lattice. In ACIRD, a query is formulated as a sequence of terms. Similarity match based on VSM is applied to decide the relevance of objects and classes. Applying the learned classification knowledge (represented by a set of keywords) of the class, collaborative two-phase search is used in the system. In the first phase (class-level search), query terms are used to search qualified classes. The qualified classes form a shrunk view of the system hierarchical lattice. In the second phase (object-level search), if the user wants to further retrieve objects in some classes, ACIRD generates a query to further retrieve the classes or documents. ACIRD presents a view as a logical lattice by integrating qualified classes and matched objects with attached MG and supports.

The following terminology is used throughout the paper. We denote an element as a lower case letter, and the set or series of the elements as an upper case letter. For example, C denotes a class, and C denotes a set of classes. Each notation represents an abstraction of entity in the system.

- ACIRD Lattice ($L_{ACIRD}(C,R)$), is a graph consists of a set of class nodes (C) and a set of relations (R) between classes in C .
- Class (C) indicates a category of L_{ACIRD} which has the semantics generalized from objects belong to C .
- Object (o) is an HTML document that consists of a series of paragraphs (pg) enclosed with HTML tags. o is an instance of one or several classes in L_{ACIRD} .
- Termt t is the word or phrase extracted from objects or generalized into classes by the learning process.
- Term t with support to an object o is defined as a word (except stop word) extracted from any sentences in an informative prargraph. The support ($sup_{t,o}$), whose value is estimated from the term frequency and the weight of the enclosed HTML tag, is used to denote the importance of t to o .
- Keyword corresponds a representative term, i.e., its information quality is better than a term.
- Memership grade (MG) is the supporting degree of a keyword to some object or class as if support is the supporting degree of a term.
- A set of selected terms in an object forms the object's Object Knowledge ($Know_o$). $Know_o$ is represented by Term Support Graph ($TSG(T,o,E)$) with directed edges in E linking terms in T to o . The edge from t_i to o is labeled with $sup_{t_i,o}$.
- By generalizing terms of objects in a class, a set of terms with supports ($sup_{t,c}$) to the class are used to represent the class's Classification Knowledge ($Know_c$). In the same way as $Know_o$, we represent $Know_c$ as a graph $TSG(T,c,E)$ with directed edges in E linking terms in T to C . The edge from t_i to C is labeled with $sup_{t_i,c}$.
- For each class C , mining association rules are applied to mine associations among terms in $Know_c$. The mined term associations form a strongly connected graph Term Association Graph ($TAG(T,E)$) which consists of a set of terms in T connected by edges labeled with the confidence ($conf_{t_i,t_j}$) of assocation between two terms.
- For each class, Term Semantics Network ($TSN(T,c,E)$) is constructed as the union of the TSG and TAG of the class c. TSN is used to represent the semantics of the class and the similarity between terms in the class.

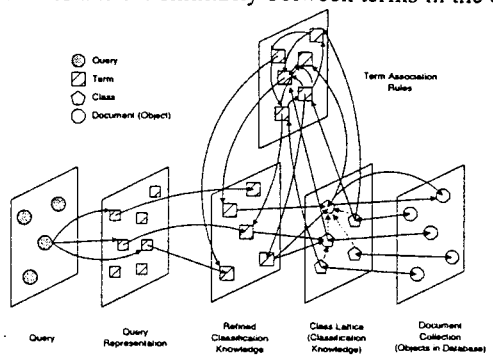

Fig 1. The overview of ACIRD.

In Fig 1, we summarize the overview of the system. First, *Document Collection* and *Class Lattice* are borrowed from Yam inventory. Then, terms with supports are extracted

from documents and generalized as *Classification Knowledge* $Know_c$. Finally, the classification knowledge is refined to a set of keywords (i.e., *Refined Classification Knowledge* $Know_c^*$) by employing mining *Term Associations* in each class. For query processing, the two-phase search module parses *Query* into term-based *Query Representation* that is used to match with *Refined Classification Knowledge* and documents term indexes.

3. THE LEARNING MODEL

In this section, we describe the learning model of ACIRD briefly. The details can be found in [10]. ACIRD applies machine learning methods to object classification by the following processes. In the training phase, ACIRD adopts supervised learning techniques and regards manually classified objects in Yam as the training set. The testing phase is described in Section 4.

3.1 Document Parsing and Terms Extraction

In ACIRD, two parsers are designed to parse the document, extract terms and calculate term supports. HTML Parser parses an HTML document into *paragraphs* enclosed by HTML tags to indicate the importance. A paragraph consists of *sentences* separated by *separators*. ACIRD categorizes HTML tags into four types: informative, skippable, uninformative, and statistical. Term Processor extracts terms in a sentence and counts the *term frequency*. According to the constructed term-base and segmentation rules of Chinese sentences, Term Processor deals with ambiguous segmentations of terms in Chinese sentence to extract terms from the sentence. After a term t is extracted from an object o, the support $sup_{t,o}$ is measured based on the definition in equation (3.1). The value indicates the importance of the term and is normalized to [0, 1].

$$sup'_{t,o} = \sum_{T_j} tf_{ij} \cdot w_{T_j},$$

where t_i is a term in the sentence enclosed by TagPattern T_j,
tf_{ij} is the term frequency of t_i in T_j, (3.1)
and w_{T_j} is the maximal weighed tag in T_j.

$$sup_{t_i,o} = \frac{sup'_{t_i,o}}{MAX_{t_i \text{ in } o}(sup'_{t,o})}, \text{ i.e., } sup \text{ is normalized to [0, 1]}$$

3.2 Feature Selection

After parsing and transforming an object into a vector of attribute-value pairs, the induction process introduced in the following section is applied to generate the classifcation knowledge. Since the complexity of the induction process is exponentially increased by the vector dimension and the noise may be increased during the induction, feature selection is needed to reduce the complexity and noise. For each object, extracted terms are filtered by a pre-defined threshold of support θ_s. Terms not being filtered out are used to represent $Know_o$. In this way, feature selection is determined by the selection of θ_s.

By analyzing the distribution of ranges of term supports shown in Fig 2, we observed that more than one half of terms whose supports are in the range [0, 0.2). Thus, in the feature selection, the system chooses $\theta_s = 0.2$ to filter out terms with low supports.

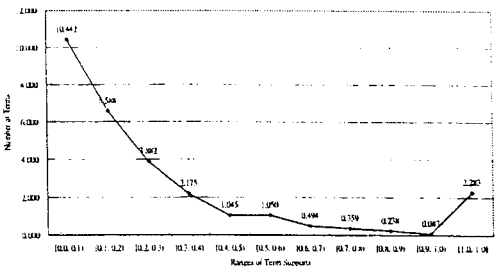

Fig 2. The range distribution of term supports of training objects.

3.3 Induction Process

The induction process is applied from the most specific classes to the most general classes. The class assignment of training objects is done manually in Yam and assumed correct. In the conventional Boolean IR systems, the term supports to an object is TRUE or FALSE, and the generalization of term t_i to class c is based on the occurrence of objects in c containing t_i. That is the generalization is according to document frequency $df_{t_i,c}$ of t_i to the class c. However, in ACIRD, the term supports to an objects is ranged from 0 to 1 rather than a Boolean value. Intuitively, it is not proper to treat the terms in an object equally significant that df_i and $sup_{t_i,o}$ of term t_i should be considered simultaneously. Thus, the support of t to c, denoted by $sup_{t,c}$, is defined in equation (3.2). Similar to (3.1), $sup_{t,c}$ is also norlmaized.

$$sup'_{t_i,c} = \sum_{o_j} sup_{t_i,o_j}, \; sup_{t_i,o_j} \text{ is the term support of } t_i \text{ to } o_j,$$

and o_j is an object in the class c.

$$sup_{t_i,c} = \frac{sup'_{t_i,c}}{MAX(sup'_{t_i,c})}, \text{ i.e., } \quad (3.2)$$

$sup'_{t_i,c}$ is normalized into $sup_{t_i,c}$ ranged [0, 1]

Likewise, terms with supports in a class form $Know_c$ as terms with supports to an object represent $Know_o$. $Know_c$ is also represented as the feature vector.

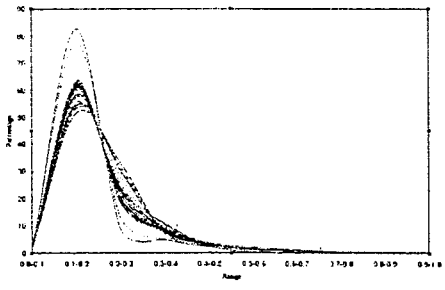

Fig 3. The distribution of term supports of all classes in ACIRD.

By analyzing the term distributions in each class, we discovered that most term supports locate in low support range (e.g., [0, 0.3)) as shown in Fig 3. If we directly filter out the class's features by a threshold θ_c, in average, less than 5% terms are left $\theta_c = 0.1$. In ACIRD, Knowledge Refining Process aims to alleviate the problem.

3.4 Knowledge Refining Process

As shown in Fig 4, if we apply the same feature selection process in Section 3.2, almost all terms in $Know_c$ will be filtered out because of very low supports. Hence, before feature selection process, we apply *mining term associations* and *perfect term support algorithm* to promote terms with low supports.

Mining Term Associations

Two important issues should be considered before mining term associations [1,2,16]. One is what granularity is used to mine associations. The other is what boundary is to generate association rules, which is corresponding to the *transaction database* defined in [2].

- The granularity of mining associations. For Web documents, the importance of a sentence of Internet document depends on the enclosed HTML tags and Internet documents are not always organized as regular documents. Therefore, we regard every informative paragraphs in an object as a transaction defined in [2].
- The boundary of generating association rules. Although the various semantics of a word is a problem for traditional data mining problem, we restrict our mining for term association only to documents in the same class rather than all the documents in the database.

Based on those reasons, we translate our problem domain into the domain of mining association rules [2] by regarding (i) *terms* are corresponding to *items*; (ii) a *document's informative paragraphs* in the class is corresponding to a *transaction*; (iii) The *class* is corresponding to the *transaction database*.

Concentrating on documents of a class instead of all classes also takes the advantage of small database size, since the complexity of mining associations is exponentially increased with the size of the database. If the size of database is not very large, a simple mining algorithm, such as Apriori [1], can be efficiently applied to our system. The definitions of *confidence* and *support*[2] [2] are modified as the follows.

$conf_{t_i->t_j} = \frac{df(t_i \cap t_j)}{df(t_i)}$, where $df(t_i)$ is the number of documents with t_i;

$support_{t_i->t_j} = \frac{df(t_i)}{N_c}$, where N_c is number of documents in class c.

[2] The support is different from the term support used in the system. It means the percentage of transactions that support the rule.

Confidence is regarded as the associative degree between two terms and employed by PTS [9] to refine $Know_c$ to $Know_c^*$. *Support* is used to evaluate the correctness of the rule and passed to a threshold to filter out noise. Based on term associations in a class, the system constructs the class's TAG. Then, TSN is generated from TSG and TAG as shown in Fig 4, i.e., $TSN(T,c,E) = TSG(T,c,E) \cup TAG(T,E)$. Nodes of TSN are also in TSG, and edges of TSN are the union of edges of TSG and TAG. Thus, the remaining task is to efficiently inference the maximal $sup_{t_i,c}^*, \forall t_i \in c$ from TSN.

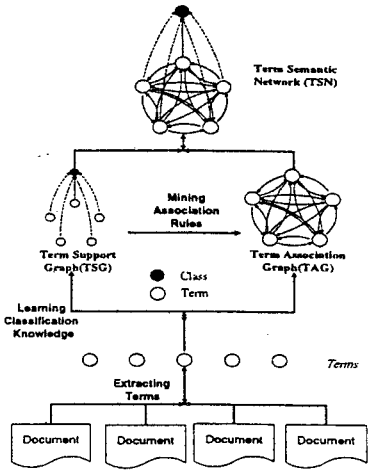

Fig 4. Construction of Term Semantic Network.

Perfect Term Support (PTS) Algorithm

As we can see, term association between terms is asymmetric. Thus, if all term associations have supports larger than a pre-defined threshold, say 0.1, TAG forms a strongly connected digraph, and so does TSN. If the system try to inference all possible paths from terms to the class, there are $n \cdot \sum_{i=1}^{n-1} P_i^{n-1}$ possible paths. Suppose a class has 10 terms. Then there are 2.3×10^8 possible paths. In ACIRD, the average number of terms of a class is 472 that the exhaustive search becomes infeasible. We propose PTS algorithm to find $sup_{t_i,c}^*, \forall t_i \in c$ in polynomial time.

Based on $sup_{t,c}^* = conf_{t,t_j} \times conf_{t_j,t_k} \times ... \times conf_{t_z,c} \times sup_{t_z,c}$, and $conf$ and sup range in [0, 1], the more edges are involved in the path from term t to class c, the smaller value $sup_{t,c}^*$ is. The proposed greedy heuristic is:

- **Heuristic**. For each term t_i associated by term t, find a term t_{max} with $sup_{t_{max},c}^* = MAX\{sup_{t_i,c}^* \mid \forall t_i \in c\}$.

The heuristic is recursively applied to the process: selecting a term with maximal support, modifying other term's $sup_{t,x}^*$, and modifying $sup_{t,c}$ as $sup_{t,c}^*$ if $sup_{t,c}^* > sup_{t,c}$. Due to the restriction of paper length, we omit the details of PTS algorithm that can be found in [9].

In [9], we have proved that PTS always finds the optimal solution in polynomial time $O(\|Know_c\|^2)$. It shows PTS can efficiently infer TSN to promote some *non-representative terms* to *representative terms* based on associations with *representative terms*.

Threshold of Support in Class

The remaining task of the Knowledge Refining Process is to select a threshold to filter out non-representative terms. An experiment is designed to compare the precision and recall of $Know_c$ and $Know_c^*$ which shows Knowledge Refining Process indeed refines $Know_c$. The experiment also shows the way to find an optimal threshold with the best compromise between precision and recall. There are two criteria to select threshold used in the experiment.

- **Top n**. The $sup_{i,c}^*$ of all terms are sorted in descendent order. The first n ranked terms are selected to be the class keywords.
- **Threshold = 0.x**. This criterion selects those terms with $sup_{i,c}^* \geq 0.x$, where $x = 1, 2, ..., 9$.

Observing the experiment results, we can find that, before applying PTS, the least precision is 0.76, because the high thresholds (Top 10, Top 20, T = 0.5, and T = 0.7) select highly informative terms only. However, as we predicted, the recall is low before PTS is applied. It implies that Induction Process can not learn the implicit association between terms, though it can generalize terms from objects to class. As for PTS, in comparison with the case without applying PTS, the precision is slightly decreased, but the recall is dramatically increased. Thus, by applying Induction Process and Knowledge Refining Process, ACIRD can dig out the hidden semantics to increase the recall rate without losing the precision. Also, among these criteria, the criterion, "Threshold = 0.5", achieves best based on the overall values of recall and precision.

Table 1. Simulation results of PTS based on precision/recall.

	Before PTS algorithm		After PTS Algorithm	
	Precision	Recall	Precision	Recall
Top 10	0.76	0.27	0.91	0.38
Top 20	0.78	0.53	0.85	0.62
Th = 0.5	0.97	0.10	0.73	0.97
Th = 0.7	0.96	0.07	0.79	0.83

4. EVALUATION OF AUTOMATIC CLASSIFICATION

In this section, we first describe our automatic classifier. Based on a series of analyses and experiments, we conclude that $Know_c^*$ is effective to be applied to automatic classification of the Internet documents.

The Classifier Based on Similarity Match

ACIRD Classifier uses the conventional similarity measurement, the cosine value of the vectorized document and class, defined in equation (4.1).

$$sim(o,c) = \frac{\sum_{t_i \text{ in } c \text{ and } o} sup_{i,o} \times mg_{i,c}}{\|o\|_2 \times \|c\|_2}, \text{ where}$$

t_i is a term shared by o and c, $sup_{i,o}$ is the support of t_i to o,

and $mg_{i,c}$ is membership grade of t_i to c, i.e., $sup_{i,c}^*$; (4.1)

$\|o\|_2 = \sqrt{sup_{1,o}^2 + sup_{2,o}^2 + ...}$ is the norm of the object;

$\|c\|_2 = \sqrt{mg_{1,c}^2 + mg_{2,c}^2 + ...}$ is the norm of the class.

Since classes are not mutually exclusive, the assignment of object to classes is not the answer of "true" or "false". For an input object, ACIRD Classifier gives the best N classes that are close the concept of the object. The classification accuracy is estimated by the criterion that if the target class of a testing object is located in the set of best N matched classes. The criterion is called "*TopN*".

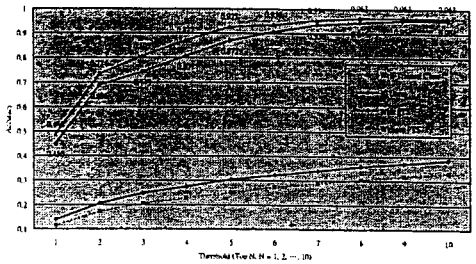

Fig 5. The experiment of classification accuracy.

In Fig 5, we compare the effective of PTS algorithm based on two curves: "Without PTS" and "With PTS". Since training sets of many classes are not enough, the classification accuracy is not well. To evaluate the quality of $Know_c^*$, keywords of each class are manually extracted by 10 human experts to form the benchmark, denoted by $Know_c^U$. The same testing is performed based on $Know_c^U$, and $Know_c^*$, which are corresponding to "10 Users" and "With PTS" according to well-trained classes. The result shows that our learning methods are capable of discovering $Know_c^*$ with quality in par with the manually extracted classification knowledge $Know_c^U$.

5. COLLABORATIVE TWO-PHASE SEARCH ENGINE

In order to reduce the overload of a single search engine, collaborative two-phase search is designed. In the first phase, ACIRD central site matches the keyword of queries with the class knowledge. Then, in the second phase, the central site updates the query and redirects to ACIRD search engines following users' feedback. The efficiency of the two-phase model depends on whether ACIRD can provide user's information need in the first phase (the class level search) or not. Thus, if the assumption that *most terms in query are located in* $Know_c^*$ is true, then class-level search based on $Know_c^*$ has the searching efficiency. We examine the assumption by the following analysis on the query log of Yam. Then, we give the

mechanism of collaborative two-phase search and some query examples.

5.1 Analyses on the Query Log of Yam

In the experiment, we analyze the Internet search query from the log of Yam collected in October 1997, and extract Chinese terms from queries to count each term's frequency. By regarding each term as an information need and its frequency as the reference count of the information need, we can discover information needs that users of Yam are interested in. There are 9644 terms, denoted by the set CT_{Yam}, with 648006 references. Then a series of tests on the distribution of keywords of $Know_c^*$ are performed to verify that the above assumption. Each test has a different threshold to select keywords for $Know_c^*$, denoted by "Th = x.x". If the selected keyword in $Know_c^*$ is the same with some term in CT_{Yam}, the reference count of the keyword is equal to that of the term; otherwise, the reference count of the keyword is 0. The summation of the reference count and the number of keywords (the *index rate*) of each test are shown in Table 2. Regarding references and query terms of Yam's query log as the based line, the recall rate and index rate of each test are defined as:

Recall rate = references of the test / references of Yam.

Index rate = indexed terms of the test / query terms of Yam.

For example, in the case "Th = 0" (i.e., no class keyword is eliminated), indexed keywords cover 96.92% information needs (the recall rate) with about double index size of query terms (the index rate). However, the case "Th = 0.5" covers 69.89% information needs with about 30.91% index size of query terms. Based on the result, we conclude the rule of thumb of ACIRD, which is similar to the 80-20 rule in relational databases.

70% queries use 30% keywords of $Know_c^*$.

Consequently, most query terms are keywords of $Know_c^*$; this result proves that *the class-level index is capable of satisfying information needs of most users*. Thus, Two-Phase Search shrinks the searching domain with little information loss to achieve the efficient search engine.

Table 2. The summation of the *recall rate*) vs. the *index rate*.

Based Line: Yam	648006 (100%)	9644 (100%)
Threshold	Recall Rate	Index Rate
Th = 0.0	628065 (96.92%)	18076 (187.43%)
Th = 0.1	477906 (73.75%)	3775 (39.14%)
Th = 0.2	470277 (72.57%)	3446 (35.73%)
Th = 0.3	465396 (71.82%)	3260 (33.8%)
Th = 0.4	458468 (70.75%)	3090 (32.04%)
Th = 0.5	452897 (69.89%)	2981 (30.91%)
Th = 0.6	440661 (68%)	2723 (28.24%)
Th = 0.7	421649 (65.07%)	2498 (25.9%)
Th = 0.8	404615 (62.44%)	2277 (23.61%)
Th = 0.9	389439 (60.1%)	2015 (20.89%)
Th = 1.0	378249 (58.37%)	1905 (19.75%)

5.2 Collaborative Two-Phase Search

In the collaborative ACIRD system, there is one ACIRD central site (class-level search engine) and several ACIRD components (object-level search engines). A global ACIRD lattice exists in the ACIRD central site and each ACIRD component owns part of the ACIRD lattice, which consists of partial classes and objects to form the shrunk view of the lattice. The architecture of collaborative ACIRD search engine is shown in Fig 6.

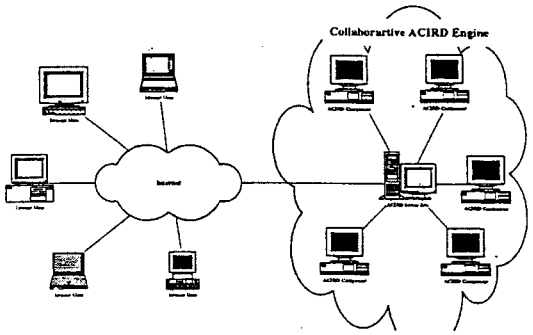

Fig 6. Architecture of ACIRD Collaborative Two-Phase Search.

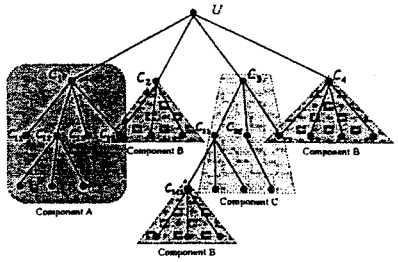

Fig 7. Global ACIRD lattice in the ACIRD Central Site.

Queries from Internet users are served by ACIRD central site to find the matched classes, and the central site determines the components involved in this query processing. Then, the second phase search is carried out by the ACIRD components. In order to illustrate the process clearly, an example of the global ACIRD lattice is shown in Fig 7.

In this example, there are four most general classes, C_1, C_2, C_3 and C_4 and several component lattices. Let L_{ACIRD} indicates the global ACIRD lattice and L^i_{ACIRD} the component lattice of ACIRD component i. The formal definition of the relationship between L_{ACIRD} and L^i_{ACIRD} is defined as follow.

$$L_{ACIRD} = L^1_{ACIRD} \cup \cdots \cup L^n_{ACIRD}$$

For instance, the global ACIRD lattice is decomposed into three components, Component A, B and C that each resides in an ACIRD component as shown in Fig 8. The lattice in Component A consists of the class node C_1 and all of its subclasses, and Component B consists of the class nodes C_2, C_4 and C_{311}, which is a subclass of class C_3. At last, class C_3 and all of its subclasses except C_{311} reside

in Component C. The allocation relation is kept by the ACIRD central site to dispatch the second phase search.

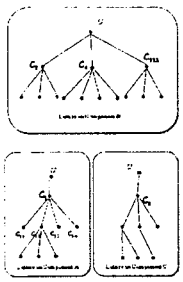

Fig 8. Lattice on ACIRD components.

5.3 Collaborative Two-Phase Search Algorithm

Based on the previous analysis of user log, it is worth to perform class-level search in the first phase to shrink the search domain. The first-phase search is capable of satisfying about 70% queries efficiently and the outcome can have a better presentation. The conventional searching approach, i.e. search for all the objects, is also implemented in the system. After the user quickly reviews the result of class-level search, he or she can choose object-level search in some classes or search all objects. The algorithm of Collaborative Two-Phase Search is described in the following and the data flow diagram is shown in Fig 9.

Two-Phase Search Algorithm

1. **Query Parsing**: Remove stopwords from query string. Parse the query string into a sequence of keywords indicated by keyword IDs (KIDs).
2. **Class-Level Search**: Retrieve classes (CID) associated to the KID, calculate the relevance score of each matched class, and sort CID by the score in decreasing order. The result presentation can be a ranked class list or a hierarchical probability-tree based on the lattice.
3. **Object Query Dispatch**: After the user selected the interesting classes, the query is decomposed and modified to the object query plan for thses class.
4. **Object Query Refinement**: the object query for each component is refined to improve the query efficiency and the precision/recall of results.
5. **Object-Level Search**: For each selected ACIRD component, retrieve objects associated to the KID, calculate each object's relevance score, and sort OID by the score in decreasing order.
6. **Query Result Collection**: Collect the results of object level search from ACIRD components, retrieve the object's information, and present the result.
7. **Search all objects**: If Two-Phase Search was unable to find the desired information (that is the 30% lost rate in the Yam query log analysis), the user can choose the conventional search approach to search all objects.

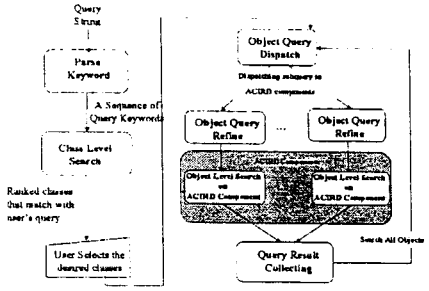

Fig 9. Data flow diagram of Two-Phase Search algorithm

ACIRD components selection

After user selects an interesting class, ACIRD must make a decision: how many components to work cooperatively to solve the query. The decision requires reasoning about *lattice location* and *ranking of the ACIRD components*. Because the lattice in each component is not mutually exclusive, a class node may exist in several ACIRD components. ACIRD central site has to find the components, which contain the interesting classes, as the candidates and generate query plans for them. For a class, if there are more than one candidate component which have redundant objects, ACIRD will choose one component with lower processing load to maximize the query parallelism to minimize the response time.

5.4 Examples of Two-Phase Search

Users can search desired objects from ACIRD[3] by giving their query strings.

For example, in Fig 10, the user selects the query mode "Two-Phase Search" and gives the query "interesting technical magazine".

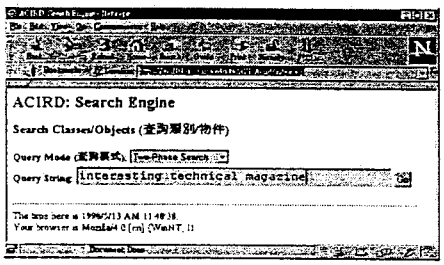

Fig 10. Two-Phase Search in ACIRD (Query Interface).

The result of the query is shown in Fig 12. "*Rank*" indicates the ranking order of the matched classes. "*Matched Class*" presents the classes matched the query. "*Object In Class*" shows two links, "All" and "Direct". "*MG*" indicates the normalized relevance score between

[3] http://YamNG.iis.sinica.edu.tw/Acird/class.htm only provides Chinese interface. The figures shown in this example are the English translations.

the query and matched classes. The page in Fig 11 shows 8 matched classes, and the user can press the link in the bottom of the page to perform the conventional search on all objects, if no classes are desired.

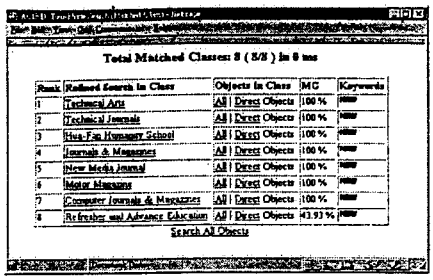

Fig 11. Query Result: Matched Classes.

If the user focuses on the class "Technical Journal" and clicks on it to perform object-level search, the search result will be refined and shown in Fig 12. "*In Class*" presents one or several classes that own the object. It can be clicked to list all objects in the class. "*Object Title*" is linked to the physical page on the Internet. "*Latest Date*" indicates the latest date that the Internet robot visited the page. "*Status*" shows the status of the latest visit.

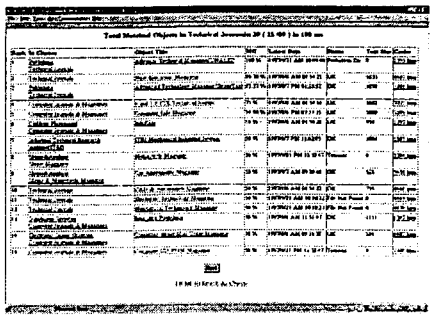

Fig 12. Query Result: Search Objects in Specific Classes.

If the user presses the link to perform "Search All Objects", the result is shown in Fig 13. There are totally 746 relevant objects in this example. In comparison with 8 relevant classes of class-level search, it is infeasible to visit and find information from these 746 links.

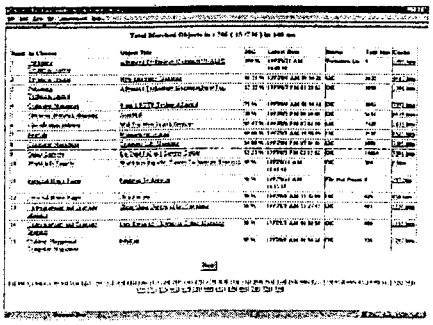

Fig 13. Query Result: Search All Objects.

6. CONCLUSIONS

In this paper, we shows that machine learning and data mining techniques can be applied to learn and refine the classification knowledge and mine the associations of terms in a specific domain (class). Based on the knowledge, automatic classification is also able to organize the Internet documents in a class hierarchy. According to the analysis of the query log of Yam, we also prove that the knowledge can be the *meta-index*, and ACIRD is capable of shrinking the searching domain efficiently based on the index. The meta-index is also useful to retrieve classes containing possibly desired documents in very short time and to present a comprehensible view of concepts in class-level search. Considering the explosive growth of Internet documents and Internet users, collaborative query archiecture can reduce the consumption of network resource and provide high performance of Internet document search.

7. REFERENCES

[1] R. Agrawal and R Srikant, "Fast Algorithms for Mining Association Rules", Proceedings of the 20th International Conference on VLDB, September 1994.

[2] R. Agrawal, T. Imielinski, and Swami, A., "Mining Association Rules between Sets of Items in Large Databases", Proceedings of the ACM SIGMOD International Conference on Management of Data, May 1993.

[3] C. H. Chang and C. C. Hsu, "Customizable Multi-engine Search Tool with Clustering", The Sixth World Wide Web Conference, April 1997.

[4] B. Yuwono, S. L. Y. Lam, J. H. Ying, and D. L. Lee, "A World Wide Web Resource Discovery System", World Wide Web Journal, Vol. 1, No. 1, Winter 1996.

[5] William Cross, "All-In-One Search Page", http://www.albany.net/allinone/, 1995.

[6] David Eichmann, "Ethical web agents," Electronic Proceedings of the Second World Web Wide Conference '94: Mosaic and the Web, 1994.

[7] W. B. Frakes and R. Baeza-Yates, "Information Retrieval – Data Structures & Algorithms", Prentice Hall, 1992.

[8] S. H. Lin, M. C. Chen, J. M. Ho, and Y. M. Huang, "The Design of an Automatic Classifier for Internet Resource Discovery", ISMIP'96, December 1996, pp. 181-188.

[9] S. H. Lin, C. S. Shih, M. C. Chen, J. M. Ho, M. T. Kao, and Y. M. Huang, "Extracting Classification Knowledge of Internet Documents: A semantics Approach", ACM SIGIR'98, 1998, pp. 241-249.

[10] S. H. Lin, C. S. Shih, M. C. Chen, J. M. Ho, M. T. Kao, and Y. M. Huang, "ACIRD: Intelligent Internet Documents Organization and Retrieval", submitted to IEEE Trans. On Knowledge and Data Engineering.

[11] G. Salton and M. J. McGill, "Introduction to Modern Information Retrieval," McGraw-Hill, 1983.

[12] G. Salton, "Automatic Text Processing", Addison Wesley, 1989.

[13] E. Selberg and O. Etzioni, "Multi-service search and comparison using the MetaCrawler", Proceedings of the 4th International World Wide Web Conference, December 1995.

[14] Erok Selberg and Oren Etzioni, "The MetaCrawler WWW Search Engine," http://metacrawler.cs.washington.edu:8080/home.html, 1995.

[15] InfiNET Services, "InfiNET META search," http://members.gnn.com/infinet/mata.htm, 1996.

[16] R. Srikant and R. Agrawal, "Mining Quantitative Association Rules in Large Relational Tables", Proceedings of the ACM SIGMOD International Conference on Management of Data, June 1996.