# ACIRD: Intelligent Internet Document Organization and Retrieval

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

## Abstract

*This paper presents an intelligent Internet information system, Automatic Classifier for the Internet Resource Discovery (ACIRD), which uses machine learning techniques to organize and retrieve Internet documents. ACIRD consists of a knowledge acquisition process, document classifier and two-phase search engine. The knowledge acquisition process of ACIRD automatically learns classification knowledge from classified Internet documents. The document classifier applies learned classification knowledge to classify newly collected Internet documents into one or more classes. Experimental results indicate that ACIRD performs as well or better than human experts in both knowledge acquisition and document classification. By using the learned classification knowledge and the given class lattice, the ACIRD two-phase search engine responds to user queries with hierarchically structured navigable results (instead of a conventional flat ranked document list), which greatly aids users in locating information from numerous, diversified Internet documents.*

**Index Terms**: Document Classification, Data Mining, Information Retrieval, and Search Engine.

## 1. Introduction

The explosive growth of the Internet has revolutionized working and living patterns as it has evolved into a major source of information and communication medium. However, the huge amount of information on the Internet has created the information overflow phenomenon. To alleviate this problem, many Internet search engines and topic directories have become are available to users. Search engines, such as AltaVista and InfoSeek, are able to retrieve Internet documents in response to a user's query. Alternatively, topic directories e.g. Yahoo!, allow users to search relevant documents by browsing a topic hierarchy. Search Engines are designed to efficiently organize and access a large collection of documents. Since the number of documents available on the Internet is huge, thousands of documents may be retrieved by a search engines for a query with one or two terms[1]. For example[2],

[1] The average query length is 1.3 terms, as reported by [37].

[2] Experiments were performed in October 1998. All the "facts" given in this paper reflect the situation then.

given the query "education and university," there were 87,368,493 hits by AltaVista, 7,379,086 hits by Infoseek, and 237,902 hits by WebCrawler. Ranking a large number of documents using very few terms is unlikely to produce an order of documents that meets the user's information needs. Consequently, the user must retrieve many uninteresting documents before obtaining the desired information. Several search engines have applied relevance feedback [32] to expand and/or refine the query based on documents selected by the user. However, relevance feedback may be ineffective since grasping the user's true intention from the selected documents is extremely difficult.

The conceptual gap between document developers and users enlarges the difference between retrieval results and user expectations. Due to the richness of language and culture on the Internet, web developers and users may use different terms and expressions to represent the same concept, or use the same term to describe different things. Therefore, term-based search engines frequently retrieve documents, virtually thousands, not desired by users, while the desired documents may not be retrieved. For instance, term-based search engines do not match the term "airline schedule" in documents with the term "flight schedule" in a query, whereas both terms are considered to have the same meaning. Consulting a thesaurus may resolve the problem. Another problem arises in which a term may have different meanings in different contexts, such as the term "bank." Building a thesaurus for each specific domain can solve this problem. However, no static thesaurus can handle the shifting semantics of terms in the Internet environment owing to its diversity and dynamic nature.

Current topic directory systems suffer from the bottleneck of manual classification of newly collected documents. For example, Yahoo!, the largest directory system on the Internet, contains roughly 1.2 million links in its topic hierarchy, and more than 150 editors are needed to classify web pages[3]. The total number of documents in the directory systems is much less than the database used by search engines. The focus of directory systems is assigning the Internet documents to the right topics, instead of the speeding up or increasing the size of the database.

From the traditional measures of information retrieval, i.e., precision and recall, search engines have low precision rates because too many results are retrieved, while topic directories suffer from low recall rates because of their small databases. To achieve balanced precision/recall rates and allow users to access needed documents rapidly, organizing documents according to a set of classes is a prerequisite for efficiently managing and retrieving Internet documents [14]. The system *ACIRD*[4] (Automatic Classifier for the Internet Resource Discovery) [21, 22, 23] was designed to achieve

[3] http://searchenginewatch.com/reports/directories.html in October 1998.

[4] http://YamNG.iis.sinica.edu.tw/Acird/class.htm

efficient and effective Internet document organization and retrieval. The system learns classification knowledge from classified documents. It also mines the association rules among terms to explore the implicit term semantics, and infers from the term associations in order to refine the classification knowledge of classes in a class lattice. To facilitate the Internet search, the system uses a two-phase search mechanism that presents a hierarchically navigable view to the user.

The rest of this paper is organized as follows. Section 2 reviews related works. Section 3 introduces the functions of ACIRD. In Section 4, we introduce the conceptual model and define the terminology used in this paper. Section 5 discusses the ACIRD learning model in detail. Next, Section 6 presents experiments on automatic classification of documents to justify the design decisions of ACIRD. Section 7 then introduces the two-phase search method. Contributions of this work and areas for future work are finally presented in Section 8.

## 2. Related Work

This section reviews works related to this study on Internet information retrieval, document classification and data mining.

### Internet Information Retrieval

Previous studies on Information Retrieval (IR) systems focused mainly on improving retrieval efficiency by using term-based indexing [8, 11, 28, 37] and query reformulation [32] techniques. Term-based document processing initially extracts terms from documents using a pre-constructed dictionary, stop words and stemming rules [10, 28, 30]. Once terms are extracted, a widely used method called TF×IDF (or its variations) [31, 33] is applied to determine the weights of terms. A document can thus be represented by a set of terms and their weights. The similarity measure between a query and a document is the direct product of their corresponding term vectors, the cosine value between the two vectors in a multi-dimensional vector space. To indicate the degree of relevance of documents and queries, retrieved documents are presented as a ranked list based on the measure.

Alternatively, the string-based indexing approach indexes strings and all possible sub-strings instead of terms as in term-based approach, which is particularly useful for arbitrary-length string search, such as string matching (e.g. address matching) and character-based language search (including many oriental languages, such as Chinese and Japanese). Notably, the storage requirement of the string-based indexing approach is much higher than that of the term-based indexing approach. In addition, their complicated data structures take more time for retrieval. While superior in retrieving

exactly matched strings, the string-based indexing approach is inappropriate for Internet information discovery tasks in which users only give conceptual descriptions, instead of exact strings. Many investigations have developed string-based indexing technologies, including PAT-tree [4] and signature files [7], to enhance the performance of various search functions, such as prefix searching, range searching, longest repetition searching, most significant and most frequent searching, regular expression searching, etc [10]. However, these search functions are rarely found on the Internet.

While current search engines employ a variety of IR techniques, the differences among them are related to indexing, representation, querying and implementation.

*Indexing.* Search engines gather Internet HTML documents (i.e. web pages) from user submissions or by means of automatic Internet robots (also called spiders or crawlers). As conventional IR systems, search engines index a set of words or phrases for efficient retrieval. Some search engines attempt to determine and index concepts in documents. For example, Excite[5] knows that a relationship exists between related concepts like "elderly people" and "senior citizens." Based on the rich format of HTML, search engines can enhance weights of terms according to the significance of tags.

*Representation.* Most search engines employ full text indexing for fast matching between queries and documents where documents are represented by a set of term-weight pairs as the case with conventional IR systems. Most topic directory systems also provide key word search functions so that they represent web pages as term-weight indexes. In addition, they store pages into a topic hierarchy developed and maintained manually.

*Querying.* Search engines employ several functions to refine the numerous search results. For example, most search engines provide Boolean operators to derive precise results. Other functions, such as exact phrase matching, sorting pages by corresponding sites and restricting search from specified sites, are also useful for refining search results.

*Implementation.* Both Internet search engines and topic directory systems need to cope with the dynamic Internet environment, in contrast with the stable context of IR systems. Web pages are created, modified and deleted frequently, which requires systems equipped with dynamic storage structures and efficient indexing mechanisms. Implementation of intelligent Internet robots is yet another challenging issue for Internet web page collection.

Currently, there are hundreds of search engines that apply the technologies of IR to the Internet. Popular search engines are famous for their rich indexes and fast response. In general, most search

[5] http://www.excite.com/info/search_help/

engines borrow indexing and ranking methods from IR and improve their performance by adding advanced hardware and sophisticated software. User satisfaction suffers not when no matched documents are returned by search engines, but when too many documents are returned. To learn more about the current status of popular web search engines, readers can refer to Search Engine Watch[6].

**Document Classification and Data Mining**

Many approaches used to classify documents can be divided into two main camps, manual classification and automatic classification. Manual classification of documents is time consuming and expensive, which makes it infeasible for handling the huge number of Internet documents. For automatic classification, classification knowledge can be acquired from domain experts or learned automatically from training documents [3]. Acquiring knowledge from domain experts, while relatively effective, is expensive in terms of time and knowledge maintenance. Furthermore, the acquired knowledge may be incomplete, which will require the use of complicated models and theories to apply it. On the contrary, classification knowledge automatically learned from training documents is efficient, but its accuracy is constrained by the employed learning model and training data.

Many text categorization studies have focused on information retrieval [3, 9, 14, 15, 16, 17, 19, 20, 35]. Herein, "document classification" instead of "text categorization" is used since this work focuses on Internet HTML [18] documents rather than general texts. Document classification involves the automatic grouping of documents. Many studies have addressed this issue by adopting similarity-based document retrieval [35], relevance feedback [32], text filtering [25], text categorization [3, 17, 20], and text clustering [12, 19]. For example, ExpNet [35] uses similarity measurement as the category ranking method to determine the best category for the input document. SIFTER [25] uses a vector space model for document representation, unsupervised learning for document classification, and reinforcement learning for user modeling to filter documents based on content and user specific interests. INQUERY [17] employs three different classification techniques: a *k*-nearest-neighbor approach using belief scores as the distance metric, Bayesian independence classifiers, and relevance feedback. Goldszmidt and Sahami proposed document clustering based on the probabilistic overlap between documents and document clusters [12].

Previous machine learning studies developed many algorithms that have been well tested and used in many fields, such as medicine and finance. Widely used algorithms, including ID3 [26], C4.5 [27],

[6] http://searchenginewatch.com/

CN2 [5], and AQ algorithm [24], have been applied to structured training data instead of non-structured textual data in the document classification problem. Correspondingly, many approaches to document classification use feature sets to characterize documents and apply algorithms such as Bayesian classifiers [19], k-nearest-neighbor method [9, 11], rule-based induction algorithms [3], and mixed approaches (e.g. INQUERY [17]) to classify documents. While concentrating on the document classification process and learning algorithms, those systems ignore the diversity of documents in the use of terms and their semantics. In many learning applications, the characterized feature is an attribute-value pair with the assumption of fixed semantics. However, the semantics of a feature varies with different domains. For example, the document feature "apple" has different meanings for the domains "computer" and "food."

Mining association rules [1, 2, 34] are applied to discover important associations among items of transactions. A conventional application of mining item associations is finding an optimal item arrangement in a supermarket so as to allow customers to gather their groceries conveniently. In this study, we apply mining association rules to explore the semantics of features in a document.

## 3. The ACIRD System

ACIRD [21 22, 23] automatically collects and classifies Internet documents for efficient, effective management and retrieval. ACIRD initially focuses on improving the expensive and time-consuming manual classification process used by many Internet search engines. By employing classification knowledge learned from manually classified Internet documents, ACIRD automatically classifies newly collected Internet documents. Classification knowledge together with the given class lattice enables two-phase searching, which presents the search results in a hierarchical view instead of a ranked document list as in conventional Internet document retrieval. All the design decisions, such as measurement metrics, aim to reach the integral goal of ACIRD, i.e. auto-learning, auto-classification and two-phase searching. This section provides an overview of ACIRD, and the subsequent sections present the details of each component.

Fig. 1 schematically shows the workflow in ACIRD. The domain expert provides a class lattice as the worldview of the document domain and a set of training data, which are Internet documents with manually assigned classes. The *Classification Learner* learns from the training data and generates the classification knowledge (or so called class indexes) of the classes in the class lattice. The *Internet Robot* automatically collects documents from the Internet, and the *Preprocessing Process* extracts the features from the documents. The *Document Classifier* proceeds to assign one or more most appropriate classes to the incoming documents. When Internet users query ACIRD, the *Two-Phase*

*Search Engine* matches the queries with knowledge of documents, and classes and presents a hierarchical view to the users to facilitate the information discovery job. In this study, we focus on Internet HTML documents only, referred to herein as *objects*. Each object is assigned a unique ID and parsed into a document index, a set of terms with weights, which is stored in the database. The set of term and weight pairs form the feature vector representing the object knowledge. Inverted indexes pointing to the occurring objects from terms are generated for efficient access during learning and accessing.

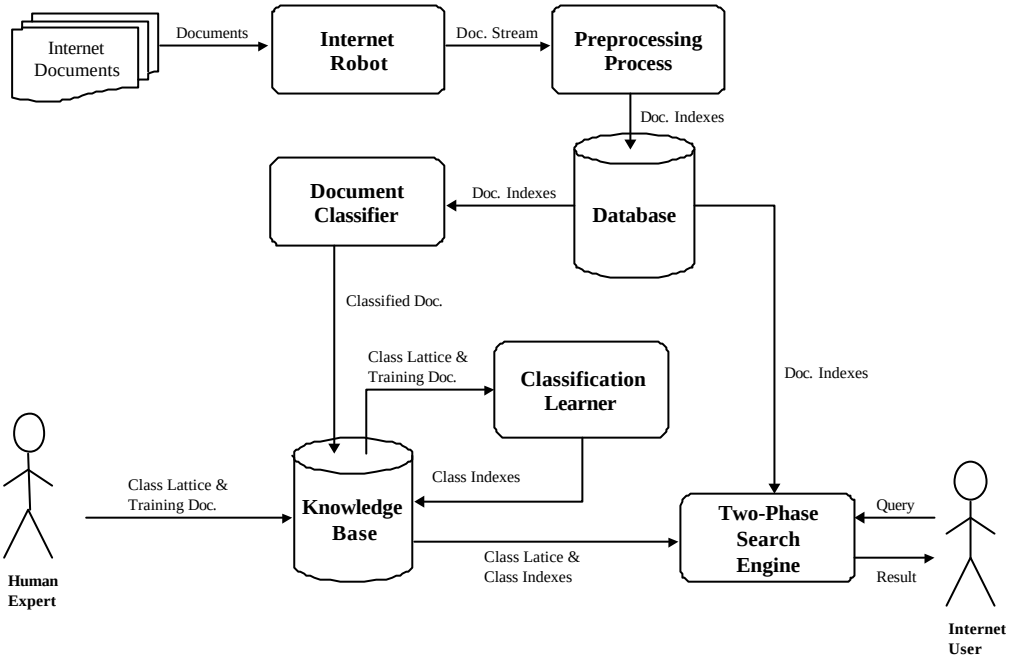



Fig. 1. The Major Components and Workflow in ACIRD.

The given class lattice presents the worldview to ACIRD. In the lattice, each node represents a class, and every parent node is a super set of its child node. Nodes with no parent nodes besides the universal node are referred to as the *most general nodes*, and nodes with no child node besides the null node as *most specific nodes*. The automatic learning and classification process of ACIRD consists of two phases: a training phase and a testing phase. In the training phase, the training data consist of a set of manually classified documents. The learning process learns the classification knowledge in the sequence from the most specific classes to the most general classes of the given class lattice. For the most specific classes, the classification knowledge is generalized from the knowledge of objects in the class. For the other classes, the knowledge is generalized from their child classes and direct objects (i.e. the objects belong to the class, but not to any of its child classes). After the knowledge of classification is learned, the technique of mining association rules is employed to discover term associations inside each class so as to enhance the classification knowledge. As term

associations highly depend on the class domain, the best scope to apply term associations to refine the classification knowledge is a single class. Our previous study [22] demonstrated that the mined term association can enhance the term semantics dramatically. In the testing phase, the classifier employs the learned classification knowledge to assign classes to the test documents (usually the newly collected documents), and the assignments are compared with the classes assigned by human experts to verify the quality of the learned knowledge.

ACIRD provides a two-phase search engine that allows users to efficiently and effectively retrieve interesting documents via interactive navigation of the returned class hierarchy, rather than via a sequence of ranked documents. During the two-phase search, each user query string is parsed and formulated as a sequence of terms, called *query term vector*. Similarity matching based on the vector space model is applied to determine the relevance between the query and the classes in the class lattice as well as stored objects. Note that both class and object knowledge are also represented as term vectors. During the first phase, a *class-level* search is performed in which the query term vector is used to determine the qualified classes that form a shrunken view of the class lattice. If the user decides to further explore a qualified class in the returned class hierarchy, the query term vector is again employed to calculate the relevance of the subclasses of the selected class. When the user decides to explore the objects in a class, which is called *object-level* search, ACIRD matches and retrieves qualified documents in the class. The two-phase search not only reduces the search domain, but also presents a hierarchical conceptual view to aid the user in locating interesting information.

## 4. ACIRD Conceptual Model

In this section, we define the terminology used herein and introduce the conceptual model of ACIRD [22]. An entity is denoted by a lower case letter, and a set or series of entities by an upper case letter. For example, let $c$ denote a class and $C$ represent a set of classes. In the following, we describe the system entities with their notations in parentheses beginning from the high level concepts and continuing to the low level ones.

- The *ACIRD Lattice* ( $L_{ACIRD}(C,R)$ ) is the given class lattice that consists of a set of classes $C$ as nodes and a set of relations $R$ as edges that connect two nodes in C. The parent node of an edge is a super set of its child node. The total number of classes is denoted by $\|L\|$.
- *Class* ( $c$ ) is a class node of $L_{ACIRD}$ that possesses knowledge generalized from the subclasses and direct objects in the class. The number of intermediate subclasses and objects is $\|c\|$.

- *Object* ($o$) is an HTML document that consists of paragraphs ($pg$) enclosed by HTML tags. An object $o$ belongs to one or several classes in $L_{ACIRD}(C, R)$.
- *Paragraph* ($pg$) consists of a series of sentences ($S$) that subsequently consists of terms. A paragraph $pg$ is informative if it is enclosed by informative HTML tags, which will be defined later.
- *Term* ($t$) is a word (excluding stop words) extracted from the sentences of an informative prargraph. Each term has a support value for the object in which it appears. The *support* ($sup_{t,o}$) of $t$ to $o$ is calculated using the term frequency and the weight of HTML tags, which quantifies the importance of $t$ to $o$.
- *Object Knowledge* ($Know_o$) is a set of selected terms ($T$) with supports for the object $sup_{t,o}$. $Know_o$ can be represented by the *Term Support Graph* ($TSG(T, o, E)$), in which each directed edge in $E$ from $t_i$ (in $T$) to $o$ has a label $sup_{t_i,o}$. The number of extracted terms in $Know_o$ is denoted by $\|Know_o\|$.
- *Classification Knowledge* of class c ($Know_c$) is a set of terms $T$ in which each term $t_i$ has a support value $sup_{t,c}$ to $c$. $Know_c$ is generalized from $Know_o$ of its direct objects and from classification knowledge of its child classes. Similar to $Know_o$, $Know_c$ can be represented as a graph $TSG(T, c, E)$, in which each directed edge in $E$ from $t_i$ (in $T$) to $c$ is labeled $sup_{t_i,c}$. The number of terms in $Know_c$ is denoted by $\|Know_c\|$.
- For each class $c$, the process of mining association rules is applied to mine associations among terms of $Know_c$. The mined rules are called *term associations*, and for each pair of terms, $t_i$ and $t_j$, there is a corresponding *confidence* ($conf_{t_i \to t_j}$). A strongly connected graph *Term Association Graph* ($TAG(T, E)$) can be generated by considering terms of $T$ as nodes and term associations as edges labeled $conf_{t_i \to t_j}$.
- For each class $c$, the *Term Semantics Network* ($TSN(T, c, E)$) is constructed as the union of $TSG(T, c, E)$ and $TAG(T, E)$. TSN is used to represent the semantics of the class and the relations among the terms in the class.

– The *Perfect Term Support* (*PTS*) algorithm [22] is applied to promote $sup_{t_i,c}$ of the edges in $TSN(T,c,E)$. The algorithm obtains an *optimal path* ($p^*_{t,c}$) from $t$ to $c$, where $p^*_{t,c}$ is a path with the maximum value among all the possible paths ($P_{t,c}$) from $t$ to $c$ in $TSN(T,c,E)$. The value associated with a path $p_{t,c}$ is the product of the confidence values of the edges in the path and the support of the term $t_Z$ at the end of the path to $c$, ($conf_{t\to t_j} \times conf_{t_j\to t_k} \times ... \times conf_{t_y\to t_z} \times sup_{t_z,c}$). The *optimal support* of $t$ to $c$ (denoted as $sup^*_{t,c}$) is defined as the value of $p^*_{t,c}$.

– A *keyword* ($k$) is a term that passes *Filtering Process*, which filters out terms based on some pre-specified conditions. For a keyword, its $sup^*_{t,c}$ is defined as the *membership grade* ($MG_{t,c}$) of $t$ to $c$. Application of PTS and Filtering Process refines $Know_c$ to produce *Refined Classification Knowledge* ($Know^*_c$). $Know^*_c$ is the knowledge base employed by the two-phase search engine and automatic classifier of ACIRD.

## 5. ACIRD Learning Model

In this section, we describe the learning model of ACIRD in detail. In the training phase, ACIRD adopts supervised learning techniques and treats previously classified documents as training objects. The testing phase is described in Section 6. ACIRD applies machine learning techniques to learn classification knowledge as shown in Fig. 2. The learning method is applied to each class of ACIRD lattice from the most specific classes to the most general ones. Each document is preprocessed into a weighted term vector. The dimension of the vector is then reduced by the Feature Selection Process in order to reduce the complexity of learning. For the most specific class, knowledge of the class $Know_c$ is generalized from the knowledge of all the objects in the class, which can be represented by the Term Support Graph (TSG). For classes other than the most specific classes, the learning process is the same except that the initial weighted term vectors originate from its subclasses and direct objects. The mining association algorithm is then applied to mine associations of terms in TSG, which can be represented by the Term Association Graph (TAG). Combining TSG and TAG derives the Term Semantic Network (TSN). TSN can be further refined to produce TSN*, i.e., $Know^*_c$ of the class. In each iteration of the refinement process, some terms may be promoted. As the promoted terms may be used to promote other terms, the promotion process is applied recursively until the stable state is reached.

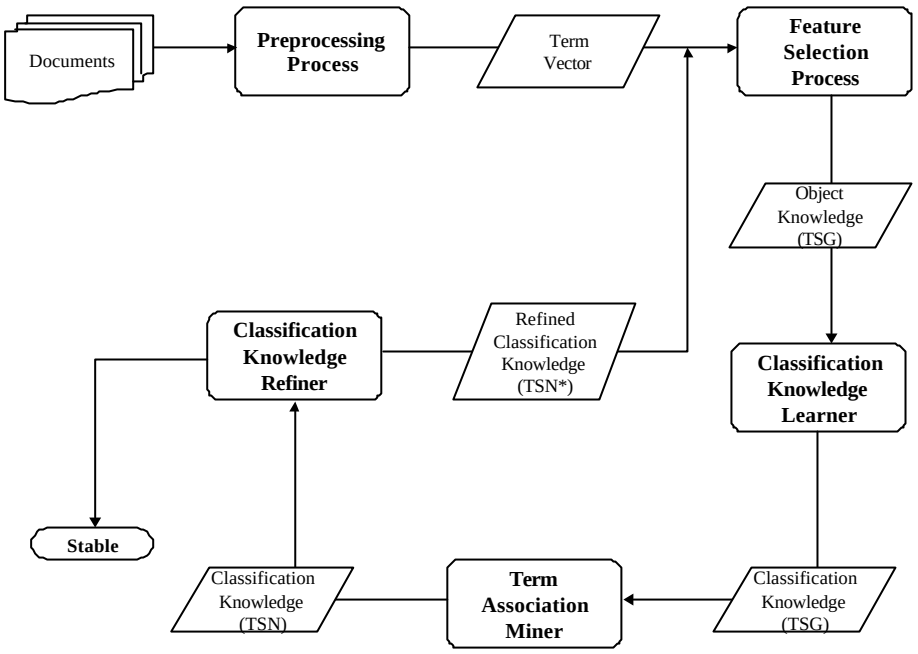



Fig. 2. The Learning Model of ACIRD.

### 5.1 Preprocessing Process and Knowledge Representation

The preprocessing process consists of two parsers, the HTML Parser and Term Parser. The HTML parser parses an object into paragraphs and determines their weights by judging their associated HTML tags. The Term Parser partitions the paragraphs into sentences and extracts terms from sentences. The Term Parser also calculates term supports using the weight assigned by the HTML Parser and the term frequency.

*HTML Parser*

An HTML document consists of paragraphs in which the associated HTML tags [18] indicate their importance and provide meta-level information. Web developers highlight the contents using HTML tags, such as titles or headings. In addition, META tags allow developers to add extra information such as "CLASSIFICATIONS" and "KEYWORDS" to the document. Apparently, the implications of tags must be considered while indexing documents. In ACIRD, human experts assign and adjust the weights of HTML tags by observing the outcomes of numerous experiments in order to improve the classification accuracy. HTML tags are classified into four types:

- *Informative.* Paragraphs enclosed by tags, such as CLASSIFICATION and KEYWORD in META, TITLE, Hn, B, I, and U, consist of either the meta knowledge of the documents or significant contents provided to users. Thus, the informative tags have the highest weights.

- *Skippable*. Tags, such as BR and P, do not affect the semantics of the document and are omitted.
- *Uninformative.* Contents enclosed by tags, such as AREA, COL, SCRIPT, and COMMENT, are invisible to users. Thus, these tags and their corresponding contents are excluded.
- *Statistical.* Contents enclosed by tags, such as !DOCTPYE, APPLET, OBJECT, SCRIPT, etc., are stored in a database for statistical purposes.

The HTML Parser is implemented with two stacks: one for HTML tags and the other for paragraphs. The algorithm is executed in one document scan with computational complexity $O(\|Know_o\|)$.

*Term Parser*

The Term Parser partitions a paragraph into sentences, extracts terms from the sentences, and counts the term frequency (TF) of each term. After a term $t$ is extracted from an object $o$, the support value $sup_{t,o}$ is measured based on TF and HTML weight, as defined in equation (5.1). This value, normalized in the range of [0, 1], indicates the importance of a term in representing the object:

$$sup'_{t_i,o} = \sum_{T_j} tf_{ij} \cdot w_{T_j}, \quad \text{where } tf_{ij} \text{ is the term frequency of } t_i \text{ in the sentence highlighted by tags } T_j, \text{ and } w_{T_j} \text{ is the maximum weighted tag in } T_j;$$

$$sup_{t_i,o} = \frac{sup'_{t_i,o}}{\underset{t_i \text{ in } o}{MAX}\{sup'_{t_i,o}\}}, \text{ where } sup \text{ is normalized to [0, 1].} \qquad (5.1)$$

Since a sentence may have more than one tag, the maximum weight of the tags is used to calculate the term support. In ACIRD, TF and the maximum tag weight are used to calculate the term support instead of using the TF×IDF weighting approach. The Inverted Document Frequency (IDF), designed to enhance the discriminating capability of high frequency terms, is not critical in our hierarchical learning model and two-phase search discovery model. In ACIRD, a high frequency term is considered to represent its class and may be generalized to classification knowledge of its parent class, instead of being used as a discriminator of objects in the class [31].

Designed to handle multi-lingual documents, ACIRD currently supports English and Chinese only. With English, each extracted term is stemmed [31]. With a character-based language like Chinese, a sentence is segmented into meaningful multi-character terms. As there may be no apparent stop characters in a sentence, the Term Parser uses a pre-constructed term base structured as a B-tree [6] to quickly match and extract meaningful terms. The Term Parser extracts Chinese terms based on the heuristics of "long term first" to resolve ambiguity. That is, for two terms, one of which is a part of the other, the Term Parser extracts the longer one. In addition, the rules for Chinese term segmentation

are provided to handle segmentations ambiguity between conflicting candidate terms. The complexity of term extraction is $O(n^2)$, where $n$ is the length of the input sentence. Including the linear time complexity of the HTML Parser, the complexity of the Preprocessing Process is $O(\|Know_o\|^2)$.

### 5.2 Feature Selection Process

After preprocessing, the object knowledge obtained is represented as a vector of attribute-value pairs, $o = \{(t_1, sup_{t_1,o}), (t_2, sup_{t_2,o}), ..., (t_n, sup_{t_n,o})\}$. Theorectically, the induction process can be applied immediately to learn classification knowledge from the object knowledge. However, the complexity of the learning process is exponentially increased by the the vector size, which requires use of the Feature Selection Process to reduce the vector size and, thus, the complexity. A common practice is to adopt a pre-defined threshold of support $q_s$ in order to discard less important terms, and the remaining terms are used to represent the object knowledge $Know_o$. In this manner, the problem of feature selection is reformulated to the problem of selection of $q_s$.

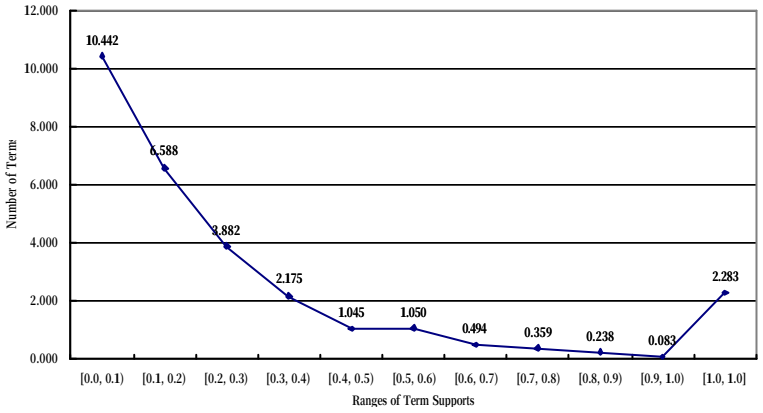



Fig. 3. The distribution of term supports of training data.

A high $q_s$ discards more terms, so the remaining terms may not be sufficient to represent $Know_o$. In contrast, a low $q_s$ only has a slight effect in the feature selection process. In ACIRD, the selection of $q_s$ is adapted to the emperical experiments. For instance, analysis of the distribution of term supports based on the training data as shown in Fig. 3 reveals that more than one-half of them are in the range [0, 0.2]. If we choose $q_s = 0.2$ as the threshold value, then the average number of terms in an object is reduced from 28.64 to 11.61, which is approximately the minimum number of terms needed to produce acceptable learning results in experiments. It is obvious that the computational cost of

feature selection is mainly due to the grouping of terms from all the objects, which can be done by means of a sorting process. Hence, the complexity is $O(\|L\| \times \|Know_c\| \times log(\|L\| \times \|Know_c\|))$.

### 5.3 Learning Classification Knowledge

The process of learning classification knowledge first generalizes the object knowledge $Know_o$ to the knowledge of most specific class $Know_c$ using induction learning, and then generalizes the obtained class knowledge to its upper classes. The learning process is applied beginning with the most specific classes and continuing to the most general classes. In conventional learning methods, the values of features of training objects are either TRUE or FALSE, i.e. whether or not the term occurs in the object; all the terms are assumed to be equally important so that the degree of term support for the object or class can be neglected. To amend this shortcoming, we define the support of $t$ to $c$, denoted as $sup_{t,c}$, in equation (5.2). Similar to (5.1), $sup_{t,c}$ is also normalized to [0, 1]:

$$\begin{aligned} &sup'_{t_i,c} = \sum_{o_j} sup_{t_i,o_j},\ sup_{t_i,o_j} \text{ is the term support of } t_i \text{ to } o_j, \text{ and } o_j \text{ is an object in the class } c; \\ &sup_{t_i,c} = \frac{sup'_{t_i,c}}{MAX\{sup'_{t_i,c}\}}, \text{ i.e., } sup'_{t_i,c} \text{ is normalized to } sup_{t_i,c} \text{ in [0, 1].} \end{aligned} \tag{5.2}$$

The support of term to a non-most specific class can be obtained from the support of term to the direct objects and child classes, as shown in Eq. (5.3). Note that the number of objects in a child class affects the contribution of that class to the super class:

$$\begin{aligned} &sup'_{t_i,c} = \sum_{o_j} sup_{t_i,o_j} + \sum_{c_j} \|c_j\| \times sup_{t_i,c_j}, \text{ where } o_j \text{ is a direct object of } c, \\ &\quad c_j \text{ is a child class of class } c, \text{ and } \|c_j\| \text{ is the number of objects in } c_j; \\ &sup_{t_i,c} = \frac{sup'_{t_i,c}}{MAX\{sup'_{t_i,c}\}}, \text{ i.e., } sup'_{t_i,c} \text{ is normalized to } sup_{t_i,c} \text{ in [0, 1].} \end{aligned} \tag{5.3}$$

The algorithm of the Classification Knowledge Learner is described in the following.

1. From the most specific classes to the most general classes, apply the preprocessing and feature selection processes to the objects.
2. For the most specific class, calculate the term support for the class based on Eq. (5.2). The computational complexity is the complexity of grouping terms from all the objects of the class, i.e. the sorting complexity, $O(\|Know_c\| \times log\|Know_c\|)$.
3. For all other classes, calculate the term support for the class based on Eq. (5.3). By regarding its child classes as direct objects, the complexity is the cost of grouping terms of child classes and

objects, $O(\|Know_c\| \times log\|Know_c\|)$. Thus, the complexity of the special-general learning process is $O(\|L\| \times \|Know_c\| \times log\|Know_c\|)$.

Due to the diversity of Internet documents, the number of terms in a class is large, and their term support is generally low. For instance, in Fig. 4, each line represents the distribution of term support of a class. From the learning results, there are about 472 terms per class on average, and support is fairly low. This figure reveals that most term supports are located in the low support range (e.g., [0, 0.3]). Therefore, a feature selection process is needed to reduce the low support terms in $Know_c$. Given a threshold $q_c = 0.1$, on average, there are 47 remaining terms per class, 24 terms with $q_c = 0.2$, and 20 terms with $q_c = 0.3$, after filtering. However, a filtering process may remove meaningful, but with low supports, terms, such as aliases, and terms closely related to high support terms. To alleviate this problem, we propose a method that uses the mining association technique to discover the term associations in a class and then apply the result to enhance supports of the otherwise filtered out terms.

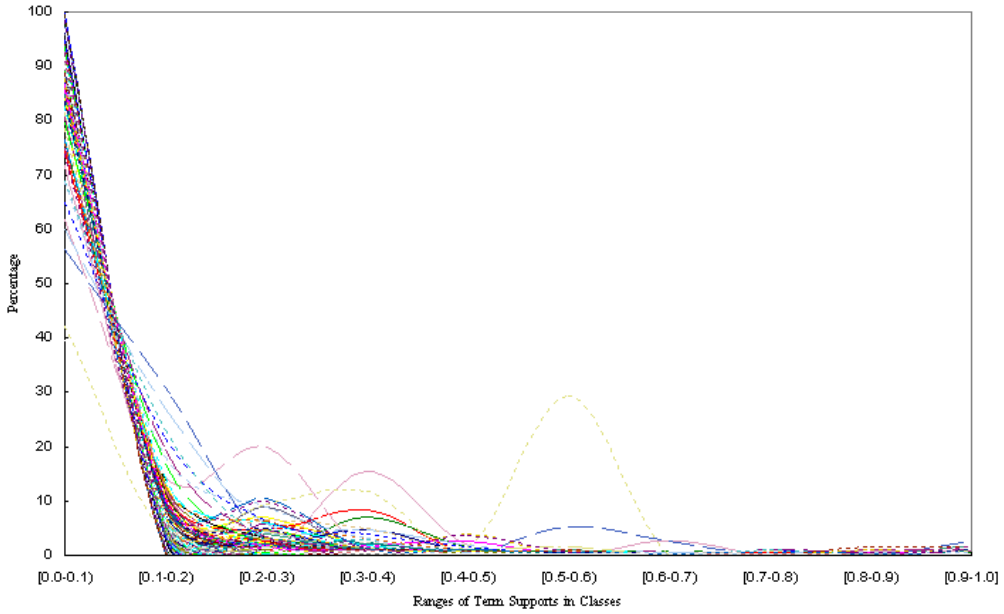



Fig.4. The distribution of term supports of all the classes from the training data.

### 5.4 Mining Term Association

The feature selection process at the class level is more sophisticated than that at the object level. First, $\|Know_c\|$ is generally larger than $\|Know_o\|$ since $Know_c$ is generalized from much classification knowledge and object knowledge. Second, terms in an object are more consistent in both semantics and representation than are those in a class. Since an object is typically written by one web developer, a simple filtering method using a threshold value can perform satisfactorily. On the other hand, the

objects in a class are collected from many web servers and written by a variety of web developers, which adds diversity to the term wording and usage. Applying a filtering algorithm that uses a threshold value $q_c$ directly to $Know_c$ may remove representative terms but with low support values. Consequently, the recall rate on $Know_c$ is likely to be low. Therefore, the system must identify and consolidate terms related to the important concepts before applying the filtering process. In ACIRD, we apply the mining term association technique and the perfect term support algorithm [22] to promote low support, representative terms. According to the association rule defined in [1], the association rule problem is defined as follows:

> Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items, and let $D$ be a set of transactions (i.e., the transaction database) in which each transaction $T$ is a set of items such that $T \subseteq I$. An association rule is an implication of the form $X \rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = f$. The rule $X \rightarrow Y$ holds in the transaction set $D$ with confidence *c* if *c%* of transactions that contain $X$ also contain $Y$. The rule $X \rightarrow Y$ has support *s* in the transaction set $D$ if s% of transactions contain $X \cup Y$.

When the above definitions are adopted, two critical issues should be addressed before implementing the data mining process: (a) the granularity of transactions used to mine associations and (b) the domain used to generate association rules.

*Granularity of mining associations*

In [13], the authors proposed restricting the granularity for generating associations to 3-10 sentences per paragraph in order to reduce the computational complexity. This restriction is impractical for web documents since a paragraph may have hundreds of meaningful sentences. In addition, the importance of a sentence in a web document depends on the associated HTML tags, not its position. Therefore, the granularity of mining term association is the whole informative paragraph.

*Domain for generating association rules*

As Internet documents are published by various web developers, it is common for a term to have different meanings that its semantics depends on both the developer and context. For example, when a document has "apple computer" in a paragraph, the semantics of "apple" are not likely to be "apple of fruit or food." Most likely, the phrase indicates "Macintosh" in the class of "Computer." Similarly, apple in "apple pie" implies the apple of fruit in the class of "Food." The above observation supports restricting the domain of mining term associations within the boundary of a class. On the other hand,

it is also common to observe a meaning has many forms of representations that make its associations a promising candidate for mining. For these reasons, ACIRD applies the mining association rule process to mine term associations based on the following assumptions:

(i) *Term* corresponds to *item*.

(ii) *Informative paragraph* corresponds to *transaction*.

(iii) *Class* corresponds to *transaction database*.

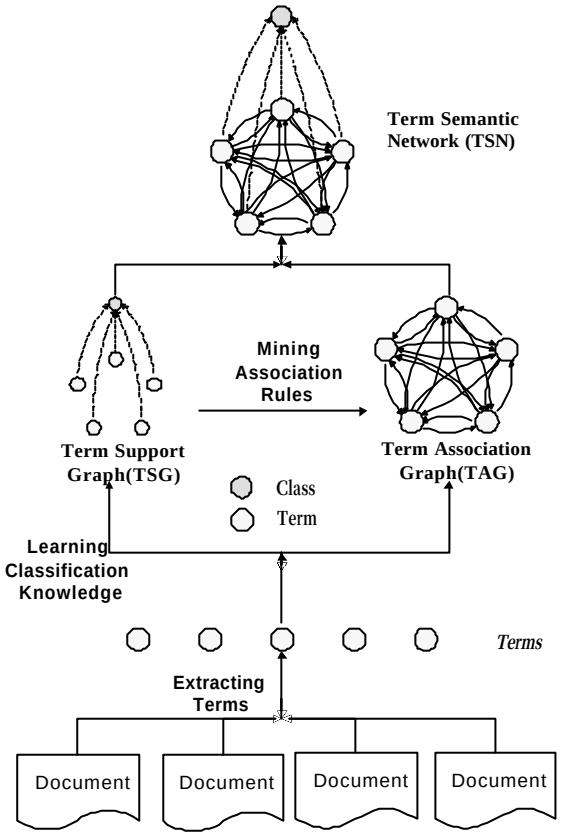



Fig. 5. Construction of the Term Semantic Network.

Concentrating on the objects of a class instead of all the classes has the merit of low computational complexity due to the resulting small database size. When the size of the database is not large, a simple mining association algorithm, such as Apriori [2], can be efficiently applied. In this study, we only consider one-to-one term associations, where the cost of the mining term associations of a class is $O(\|Know_c\|^2)$ [22].

We define *confidence* (*conf*) and *support* (*sup*) of term association $t_i \rightarrow t_j$ as follows:

$conf_{t_i->t_j} = df_c(t_i \cap t_j) \Big/ df_c(t_i)$, where $df_c(t_i)$ stands for the number of documents that contain term $t_i$, and $df_c(t_i \cap t_j)$ indicates the number of documents that contain $t_i$ and $t_j$;

$sup_{t_i \to t_j} = \frac{df_c(t_i \cap t_j)}{\|D_c\|}$, where $\|D_c\|$ stands for the number of documents in class *c*.

*Confidence* is considered to be the degree of association between terms and is employed by the Classification Knowledge Refiner to refine $Know_c$ to $Know_c^*$. *Support* is the percentage of transactions supporting the associated rules, and is considered to be a metric of the correctness of the rules. For example, $Know_c$ of class *Art* contains the following term supports: $sup_{exhibition,Art} = 0.13$ and $sup_{art,Art} = 1$. $t_{exhibition}$ may be filtered out from $Know_c$ due to its low support value. After mining the term associations of the class *Art*, ACIRD identifies the term association $exhibition \to art$ with $conf_{exhibition \to art} = 0.826$ and $sup_{exhibition \to art} = 0.1$. Assume that a rule with 10% support is considered to be useful. Following the definition of $sup^*$ defined in the previous section, $sup^*_{exhibition,Art}$ is increased from 0.13 to 0.826 (i.e., $sup^*_{exhibition,Art} = conf_{exhibition \to art} \times sup_{art,Art} = 0.826 \times 1 = 0.826$). The inference process promotes the support value of $t_{exhibition}$ to 0.826 to pass through the filter.

After the term associations of a class are mined, TSN is obtained, as shown in Fig. 5. TSG denotes the term supports of a class, TAG represents the term associations in a class, and TSN represents the union of TSG and TAG, i.e., $TSN(T,c,E) = TSG(T,c,E) \cup TAG(T,E)$.

### 5.5 Refinement of Classification Knowledge

As the term associations are asymmetric and exist for every pair of terms, both TAG and TSN form strongly connected digraphs. To determine $sup^*$ of a term, all the possible paths from the term to the class node must be considered. For a TSN, the total number[7] of possible paths from terms to the class is $n \cdot \sum_{i=1}^{n-1} P_i^{n-1}$, where *n* is the number of terms. In the experiments, the average number of terms for a class was found to be 472, which makes exhaustive search infeasible. Although the support value of term association can be employed as a filter to remove rarely used terms, it is still computationally expensive for a small number of terms. For instance, a class with ten terms creates about $2.3 \times 10^8$ possible paths. Therefore, an efficient algorithm is deemed necessary.

---

[7] $n \cdot \left[ (n-1)! + \frac{(n-1)!}{1!} + \frac{(n-1)!}{2!} + \ldots + \frac{(n-1)!}{(n-2)!} \right] = n \cdot \sum_{i=1}^{n-1} P_i^{n-1}$, where $P_i^{n-1} = \frac{(n-1)!}{(n-1-i)!}$.

Herein, we present a novel PTS algorithm to locate $sup^*_{t_i,c}$'s for all the terms in a class in polynomial time. According to the definition $sup^*_{t,c} = MAX\{conf_{t\to t_j} \times conf_{t_j\to t_k} \times ... \times conf_{t_y\to t_z} \times sup_{t_z,c}\}$, where $conf$ and $sup$ range in [0, 1], a larger number of edges in the path $p(t, t_j, t_k, ..., t_y, t_z, c)$ implies a smaller value for their product. Restated, a sub-path of an optimal path must be an optimal path as well. The proposed greedy heuristics and algorithm are as follows:

***Heuristics***. Divide the terms in TSN into two groups $T$ and $T^*$. Initially, $T$ contains all the terms, and $T^*$ is empty. Each time, find a term $t$ with the maximum $sup^*_{t,c}$ in T and move $t$ from $T$ to $T^*$. Repeat the above until $T$ is empty.

*Perfect Term Support (PTS) Algorithm*

**1.** [*Initial state*: This step initializes all $sup^*_{t_j,c}$ and partitions the terms into two groups: $T^*$ contains the term with maximum $sup^*$, and $T$ contains all others.]

Let $sup^*_{t_j,c} \leftarrow sup_{t_j,c}, \forall t_j \in c;$

Let $sup^*_{t_{last},c} \leftarrow MAX\ \{sup^*_{t_i,c} \mid \forall \mathrm{t_i} \in \mathrm{c}\};$

Let $T^* \leftarrow \{(t_{last}, sup^*_{t_{last},c})\}; T \leftarrow c - \{t_{last}\};$

**2.** [This step updates every $sup^*_{t_j,c}$ in $T$ if necessary, where $t_{last}$ denotes the latest term added into $T^*$.]

If $T$ is not empty, continue with Steps 2 and 3. Otherwise, stop.

For each $t_j \in T$ such that edge$(t_j, t_{last}) \in E$,

If $conf_{t_j\to t_{last}} \times sup^*_{t_{last},c} > sup^*_{t_j,c}$, then $sup^*_{t_j,c} \leftarrow conf_{t_j\to t_{last}} \times sup^*_{t_{last},c}$.

**3.** [This step locates the term with maximum $sup^*_{t,c}$ from $T$ and inserts it into $T^*$.]

Let $t_{last} \leftarrow \{t_k \mid t_k \in T,$ and $sup^*_{t_k,c} = MAX\ \{sup^*_{t_j,c} \mid t_j \in T\}\};$

$T \leftarrow T - t_{last};\ T^* \leftarrow T^* + \mathrm{t_{last}};\quad Output(\mathrm{t_{last}}, sup^*_{last,c}).$

In [22], we proved that the PTS algorithm always obtains the optimal solution with computation complexity $O(\|Know_c\|^2)$. PTS can efficiently promote some *non-representative* terms by exploring their associations with *representative* terms. Fig. 6 illustrates the effect of PTS. On the left-hand side, there are four *non-representative* terms in TSN. After refinement using PTS, on the right-hand side of the graph, three terms are promoted to be *representative* for their associations with the *representative* terms. All other non-representative terms and corresponding associations are eliminated to reduce the learning complexity.

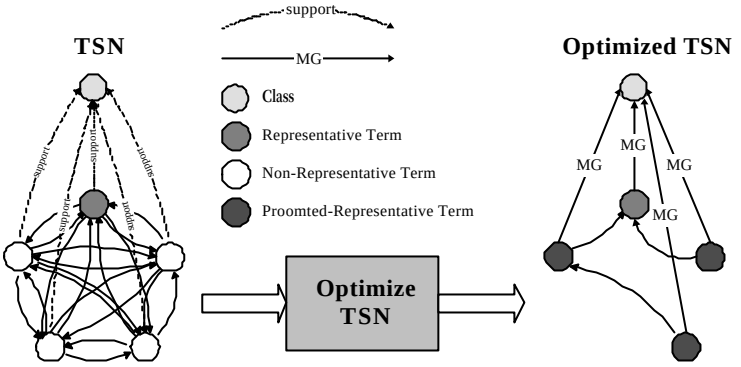



Fig. 6. PTS Refinement on TSN.

*Effects of the Knowledge Refinement Process*

An experiment was designed to compare $Know_c$ and $Know_c^*$ based on $Know_c^U$, which includes class keywords selected by ten human experts. For every class, ten human experts selected terms considered representative of the class from a given set of Internet documents. Those selected terms were sorted based on the number of times they were selected and used as the basis for judging the quality of learned knowledge. The precision and recall rates are defined as follows:

$$\text{Precision of } Know_c = \left\| Know_c \cap Know_c^U \right\| \Big/ \left\| Know_c \right\|;.$$

$$\text{Recall of } Know_c = \left\| Know_c \cap Know_c^U \right\| \Big/ \left\| Know_c^U \right\|.$$

The experiment results indicate that the Knowledge Refining Process indeed refines the knowledge contents of classification knowledge. The trade-off between precision and recall based on different feature selection criteria was examined as well. Two types of criteria were used to evaluate the outcomes of the experiment results.

- **Top n**. All the $sup_{t,c}^*$ are sorted in a descending order. The first *n* terms are selected as keywords of $Know_c^*$.

- **Threshold = ?** . This criterion is used to select terms with $sup_{t,c}^* \geq q$.

Table 1 summarizes the experiment results. Before PTS was applied, the lowest precision was 0.76, due to the high selection standards. However, the recall was low for the same reason. This observation implies that the Induction Process does not learn the implicit association among terms although it correctly generalizes the knowledge of objects to classes. In contrast to the case where PTS was not applied, PTS increased both precision and recall for the Top n criterion as it promotes

important but non-representative terms at the expense of removing less important terms (as the Top n criterion selected a fixed number of terms). For the "Threshold = ?" criterion, PTS dramatically increased recall while decreasing precision since it increased the number of keywords while promoting the terms. The experimental results confirm that the Induction Process and Knowledge Refining Process discover the hidden semantics among terms. Our results further demonstrate that an acceptable compromise between precision and recall can be achieved with a carefully chosen selection criterion.

Since each component of the ACIRD learning model can be executed in polynomial time and knowledge refinement can be achieved in a finite number of iterations, the total complexity is also polynomial.

Table 1. The performance of the PTS algorithm on classification knowledge.

| Selection Criterion | 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 |
| Threshold = 0.5 | 0.97 | 0.10 | 0.73 | 0.97 |
| Threshold = 0.7 | 0.96 | 0.07 | 0.79 | 0.83 |

## 6. Evaluation of ACIRD Automatic Classification

In this section, we describe the testing phase of the learning process. Based on the learned knowledge, the *ACIRD Classifier* automatically categorizes newly collected Internet objects. For each object, the classifier assigns one or more classes, which are compared with the classes assigned by human experts to evaluate the classification accuracy. A series of experiments and analyses revealed that $Know_c^*$ provides high quality suggestions in classifying Internet documents.

**Similarity Measurement**

$sim(o,c) = \dfrac{\sum_{t_i \text{ in } c \text{ and } o} (sup_{t_i,o} \times mg_{t_i,c})}{\|o\|_2 \times \|c\|_2}$, where $t_i$ is a common term of $o$ and $c$, $sup_{t_i,o}$ is the support of $t_i$ to $o$, and $mg_{t_i,c}$ is membership grade of $t_i$ to $c$, i.e., $sup^*_{t_i,c}$; (6.1)

$\|o\|_2 = \sqrt{sup^2_{t_1,o} + sup^2_{t_2,o} + \ldots}$ is the norm of the object;

$\|c\|_2 = \sqrt{mg^2_{t_1,c} + mg^2_{t_2,c} + \ldots}$ is the norm of the class.

The *ACIRD Classifier* uses the conventional similarity measurement, the cosine value of feature vectors of documents and classes, defined in equation (6.1):

Owing to the imprecise nature of the class concept, the class assignment of an object cannot be exactly "true" or "false." In addition, categorizing an object to one class will be impractical since an object may be conceptually related to several classes. Therefore, for an input object, the ACIRD Classifier gives the best N classes that are closest to the intention of the object. Classification accuracy is estimated based on the criterion in which the expert-assigned class of a testing object is located in the set of best N matched classes.

**Experiment Results**

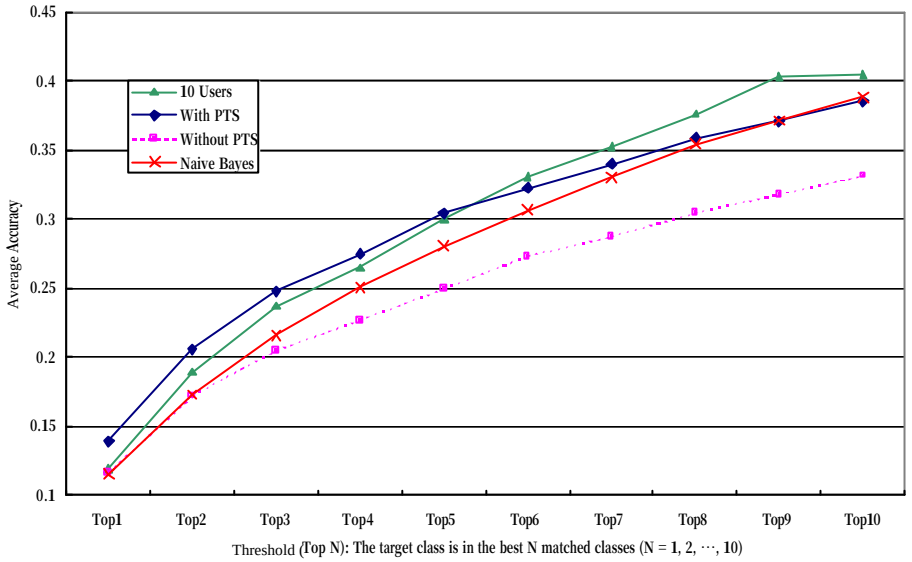



Fig. 7. The classification accuracy of assigning 8,855 testing objects to 386 most specific classes.

This experiment used 512 classes in $L_{ACIRD}$ with 386 most specific classes, 9,778 training objects and 8,855 testing objects, where both sets of objects were disjoint and manually classified. Before the learning process, ten human experts extracted the keywords from each class for use as the classification knowledge benchmark, denoted as $Know_c^U$. Fig. 7 summarizes the results of the testing processes run on $Know_c^U$, $Know_c^*$, and $Know_c$, denoted as "10 Users," "With PTS," and "Without PTS," respectively. The result obtained from a naive Bayes classifier, denoted as "Naive Bayes," was also included for comparison, as it is widely used in text classification. According to these results, $Know_c^*$ has quality on a par with the manually extracted classification knowledge $Know_c^U$ in terms of the accuracy of class assignment of objects, while both perform better than the naive Bayes classifier.

However, the classification accuracy for all the cases is unsatisfactorily low. Closely examining the training and testing sets reveals that many classes contain insufficient training objects, and that some training and testing objects contain very few keywords because they are non-text pages or link-only pages. Thus, another experiment was designed to circumvent this situation. The same testing process was performed based on the twelve most general classes[8] of $L_{ACIRD}$, and the resulting classification accuracy is shown in Fig. 8. The "Top 1" accuracy of $Know_c^*$ increased from 0.139 to 0.595. This increase was due to the sufficient number of training objects in the testing classes, and the total number of testing classes was reduced from 512 to 12.

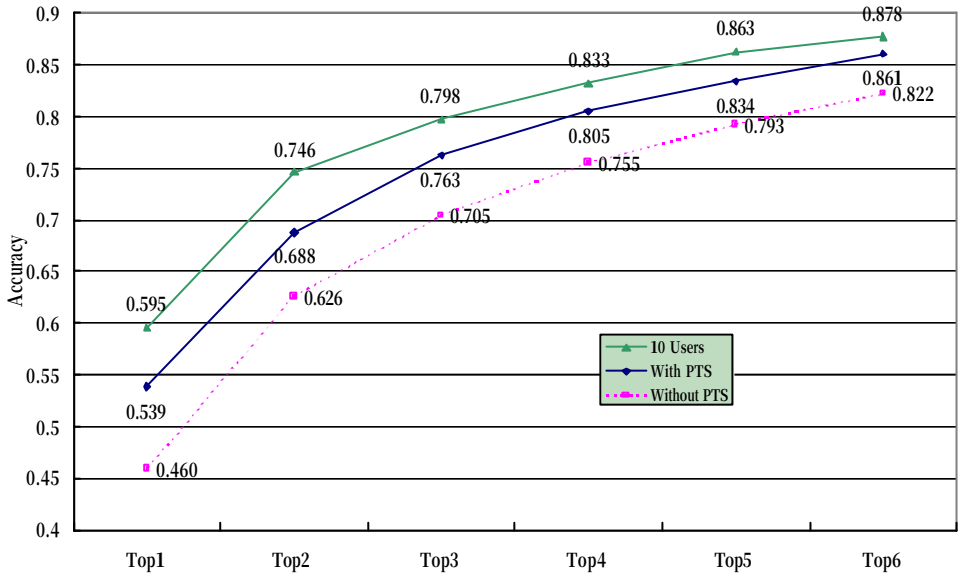



Fig. 8. The classification accuracy of assigning 8,855 testing objects to the 12 most general classes.

Table 2 lists the number of objects and keywords of the twelve most general classes. From this table, the distribution of the numbers of objects is skewed, and some classes still exhibit the problem of insufficient training objects and keywords. To further investigate this problem, we performed another set of experiments on classes with a sufficient number of training objects and keywords only. In the experiment, every testing class had at least forty training objects. The number of testing classes was reduced from 512 to 48. Since the classes contained a sufficient number of training objects, these classes are referred to herein as well-trained classes and their refined classification knowledge as well-trained classification knowledge. In Fig. 9, the "Top N" classification accuracy is markedly higher. This figure also reveals that, when the classes contained a sufficient number of training objects, our learning model could learn more accurate classification knowledge than could the human

[8] They are "Arts," "Humanities," "Social Sciences," "Society and Culture," "Natural Sciences," "Computer and Internet," "Health," "News and Information," "Education," "Government and State," "Companies," and "Entertainment and Recreation."

experts. Intuitively, the number of classes is a factor that affects classification accuracy. However, according the results shown in Fig. 8 and 9, the accuracy of classification of the twelve most general classes and forty-eight well-trained classes were comparable. For instance, Top 1 accuracies were 0.539 and 0.486, and Top 6 were 0.861 and 0.936, respectively. This result is interesting since it shows that the number of classes is not the only major factor.

Table 2. The distribution of training objects in the most general classes.

| Class Name | Objects | Keywords |
|---|---|---|
| Companies | 2702 (27.88%) | 950 (22.83%) |
| Entertainment and Recreation | 2577 (26.59%) | 1084 (26.05%) |
| Computer and Internet | 1199 (12.37%) | 471 (11.32%) |
| Education | 1169 (12.06%) | 589 (14.15%) |
| Society and Culture | 502 (5.18%) | 226 (5.43%) |
| Government and State | 384 (3.96%) | 241 (5.79%) |
| News and Information | 288 (2.97%) | 162 (3.89%) |
| Health | 280 (2.89%) | 180 (4.32%) |
| Arts | 223 (2.30%) | 115 (2.76%) |
| Social Science | 208 (2.15%) | 92 (2.21%) |
| Natural Science | 106 (1.09%) | 44 (1.06%) |
| Humanities | 53 (0.55%) | 8 (0.19%) |

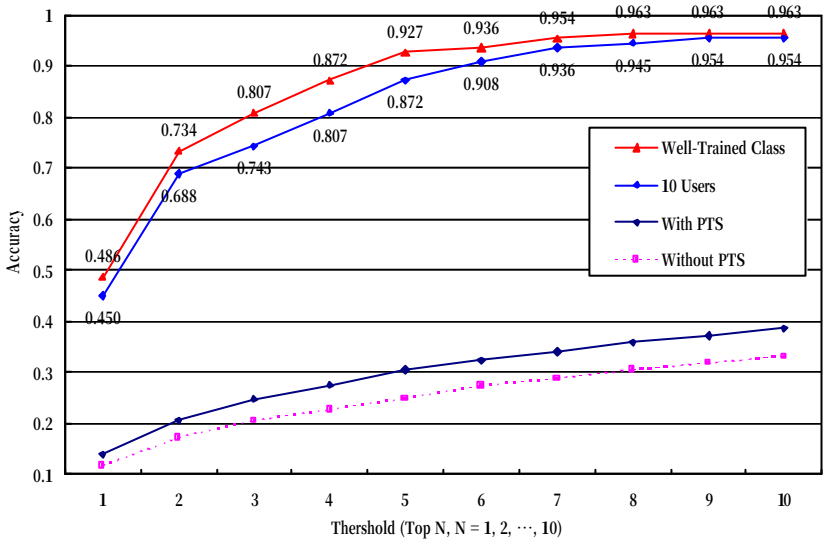



Fig. 9. The classification accuracy of well-trained classes.

## 7. Two-Phase Search Engine

Most search engines reply to user queries with a list of ranked documents, which makes it time consuming and inconvenient for users to retrieve the needed documents. ACIRD provides the two-phase search that allows users to perform both class-level searching and object-level searching. By utilizing the two-phase search, users can relate their information needs to the classes in $L_{ACIRD}$,

navigate among the classes, and locate and retrieve needed documents. The above procedure can be executed iteratively until the user obtains the needed information. Ensuring the effectiveness of the two-phase search requires that the terms of user queries be in some $Know_c^*$; thereby, applying class-level search to $Know_c^*$ can return relevant classes. The conjecture "most query terms are in $Know_c^*$" was investigated by means of the following analysis of the user query log.

**Analysis on User Query Log**

Internet queries were analyzed based on a user log collected in October 1997. Terms were extracted from queries, and their frequencies were counted. There were 9,644 distinct terms in the user log, denoted by the set $CT_{\log}$, in 648,006 queries. If a keyword in $Know_c^*$ was also in $CT_{\log}$, its reference count was assigned as the reference count of the term in $CT_{\log}$; otherwise, the reference count of the keyword was 0. The reference count is applied to measure the recall rate. The number of retained keywords of $Know_c^*$ served as a measure of the index rate, i.e. the rate of coverage of $Know_c^*$ to $CT_{\log}$. Regarding the references and query terms of the query log as the baseline, the recall rate and index rate are defined as follows:

*Recall rate* = total reference count of $Know_c^*$ / total references of $CT_{\log}$;

*Index rate* = number of keywords in $Know_c^*$ / number of query terms in $CT_{\log}$.

Table 3. Total reference counts (*recall rate*) vs. the number of keywords (*index rate*).

| Based Line: Query Terms in User Log | 648006 (100%) | 9644 (100%) |
|---|---|---|
| Filter Threshold for $Know_c^*$ | Total References Needs | Number of Keywords |
| 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 (68.00%) | 2723 (28.24%) |
| Th = 0.7 | 421649 (65.07%) | 2498 (25.90%) |
| Th = 0.8 | 404615 (62.44%) | 2277 (23.61%) |
| Th = 0.9 | 389439 (60.10%) | 2015 (20.89%) |
| Th = 1.0 | 378249 (58.37%) | 1905 (19.75%) |

To observe the change of the recall and index rates, the keyword selection process uses different thresholds denoted by "Th = x.x." Table 3 lists the reference counts (recall rate) and the number of keywords (index rate) for each test. From the table, in the case of "Th = 0" (i.e., no keyword was

eliminated), indexed keywords covered 96.92% of the information needs with about double the index size of query terms. When "Th = 0.5", the remaining keywords covered 69.89% of the information needs with an index rate of 30.91%. With a sufficiently high recall rate, two-phase search is able to shrink the searching domain to a reduced class lattice for efficient and effective searching without compromising user's information needs.

**Two-Phase Search Method**

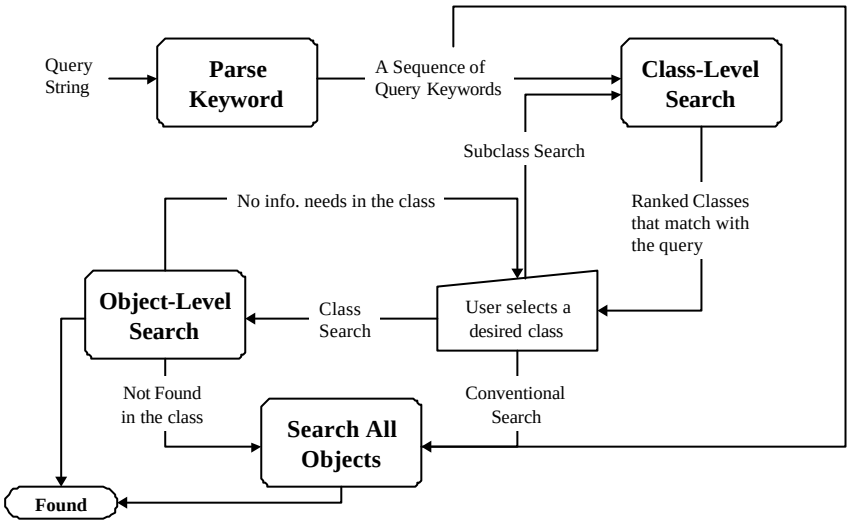



Fig. 10. Processing Flow of Two-Phase Search.

The above analyses show the likelihood of performing a class-level search without much information loss. Besides class-level searching on a structured presentation, ACIRD also provides conventional searching, including object-level searching and searching of all-objects as an "escape" for users. After briefly reviewing the return of a class-level search, the user can navigate down or up along the class lattice or choose to apply object-level search to a particular class. Fig. 10 shows a block diagram of Two-Phase Search. The operations are described below.

1. **Process query string**: Parse the query string into a sequence of keywords.
2. **Perform a Class-Level Search**: Examine the sub-classes of the designated class (initially, the root), calculate the relevance scores and sort the classes based on the scores in descending order. Return only the more general classes (i.e., if both parent and child classes satisfy the query, return only the parent class) to the user. Generate and present the result in HTML format.
3. **Execute an Object-Level Search in a class**: Retrieve objects in the designated class associated with the query terms, calculate each object's relevance score, and sort the objects according to the scores in descending order. Generate and present the findings in HTML format.

4. **Search all objects**: Retrieve all objects related to query terms, calculate each object's relevance score, and sort the objects based on the scores in decreasing order. Present the findings in HTML format.

### Examples of Two-Phase Search

The user gives the query "interesting technical magazine" to search for desired objects using Two-Phase Search.

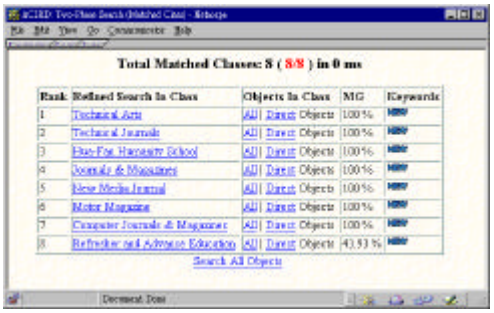



Fig. 11. Class-level Search (Query Result: Matched Classes).

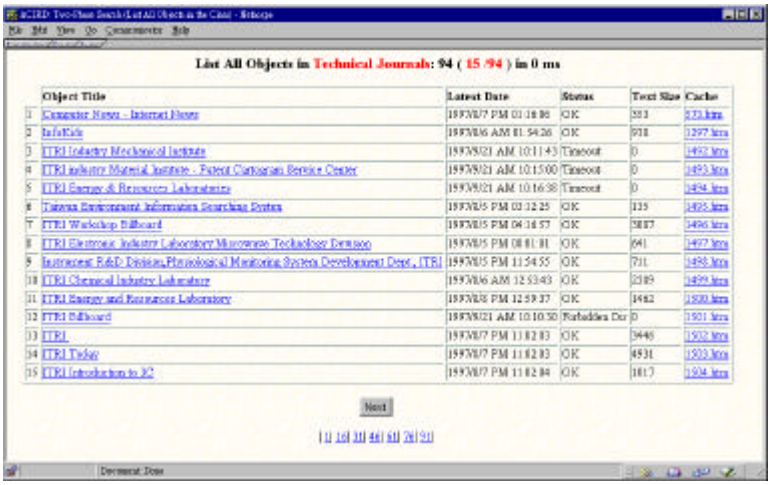
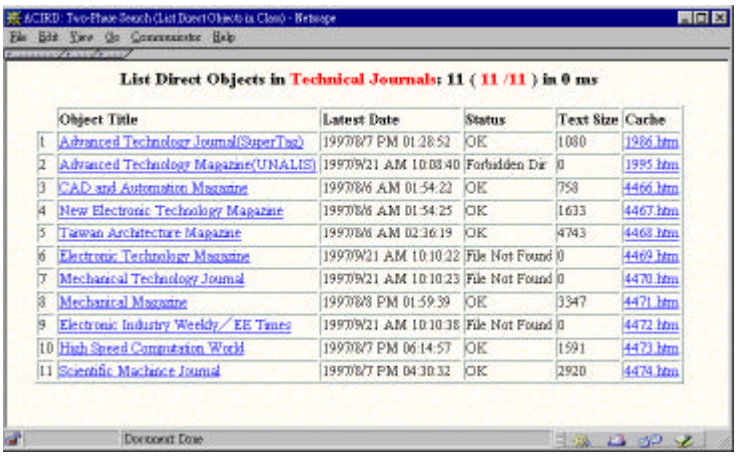



Fig.12. Search All Objects in a Class (Query Result: All the objects in a class).

Fig. 11 presents the query returns, which are the 8 highest level classes satisfying the query, obtained by class-level search. In the figure, the column "*Refined Search in Class*" presents the class names that user can resume the same query on the class by clicking on the class name. The column "*Object In Class*" shows two links, "All" and "Direct." The former lists all the objects in the class (including the objects in its subclasses), as shown in Fig. 12, which are all the objects in the class *Technical Journals*; the latter lists the class's direct objects only, as shown in Fig. 13. "*MG* (*membership grade*)" indicates the normalized relevance score.

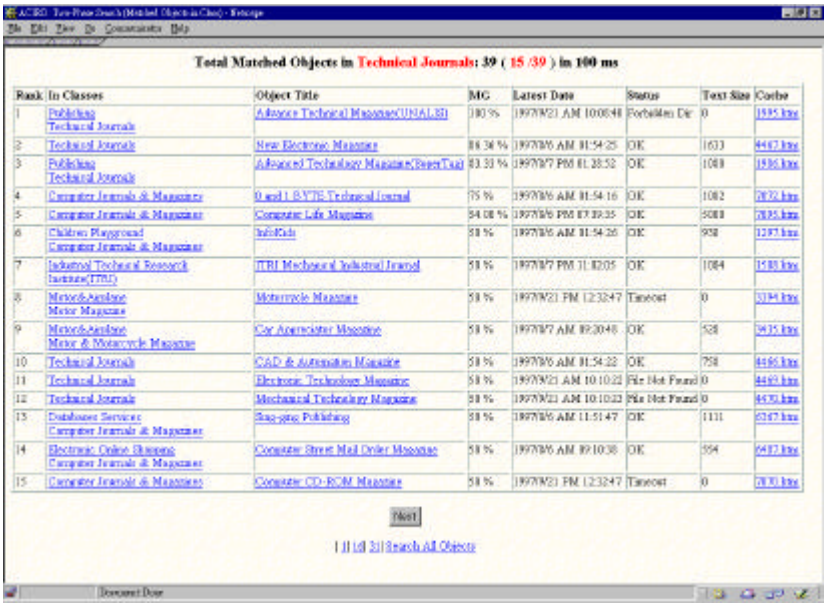


List Direct Objects in Technical Journals: 11 ( 11 /11 ) in 0 ms

| | Object Title | Latest Date | Status | Text Size | Cache |
|---|---|---|---|---|---|
| 1 | Advanced Technology Journal(SuperTag) | 1997/8/7 PM 01:28:52 | OK | 1080 | 1986.htm |
| 2 | Advanced Technology Magazine(UNALIS) | 1997/9/21 AM 10:08:40 | Forbidden Dir | 0 | 1995.htm |
| 3 | CAD and Automation Magazine | 1997/8/6 AM 01:54:22 | OK | 758 | 4466.htm |
| 4 | New Electronic Technology Magazine | 1997/8/6 AM 01:54:25 | OK | 1633 | 4467.htm |
| 5 | Taiwan Architecture Magazine | 1997/8/6 AM 02:36:19 | OK | 4743 | 4468.htm |
| 6 | Electronic Technology Magazine | 1997/9/21 AM 10:10:22 | File Not Found | 0 | 4469.htm |
| 7 | Mechanical Technology Journal | 1997/9/21 AM 10:10:23 | File Not Found | 0 | 4470.htm |
| 8 | Mechanical Magazine | 1997/8/8 PM 01:59:39 | OK | 3347 | 4471.htm |
| 9 | Electronic Industry Weekly／EE Times | 1997/9/21 AM 10:10:38 | File Not Found | 0 | 4472.htm |
| 10 | High Speed Computation World | 1997/8/7 PM 06:14:57 | OK | 1591 | 4473.htm |
| 11 | Scientific Machince Journal | 1997/8/7 PM 04:30:32 | OK | 2920 | 4474.htm |

Fig.13. Object-level Search (Query Result: Direct objects of a class).

If the user was interested in the class "Technical Journal" and clicked on it to perform object-level search, the search findings were those shown in Fig. 14. The column "*In Class*" presents the classes the object belongs to that the class hyperlink links back to the class node.



Total Matched Objects in Technical Journals: 39 ( 15 /39 ) in 100 ms

| Rank | In Classes | Object Title | MG | Latest Date | Status | Text Size | Cache |
|---|---|---|---|---|---|---|---|
| 1 | Publishing<br>Technical Journals | Advance Technical Magazine(UNALIS) | 100 % | 1997/9/21 AM 10:08:40 | Forbidden Dir | 0 | 1995.htm |
| 2 | Technical Journals | New Electronic Magazine | 88.38 % | 1997/8/6 AM 01:54:25 | OK | 1633 | 4467.htm |
| 3 | Publishing<br>Technical Journals | Advanced Technology Magazine(SuperTag) | 83.33 % | 1997/8/7 PM 01:28:52 | OK | 1080 | 1986.htm |
| 4 | Computer Journals & Magazines | 0 and 1 BYTE Technical Journal | 75 % | 1997/8/6 AM 01:54:16 | OK | 1082 | 7832.htm |
| 5 | Computer Journals & Magazines | Computer Life Magazine | 64.08 % | 1997/8/6 PM 07:39:35 | OK | 5088 | 7835.htm |
| 6 | Children Playground<br>Computer Journals & Magazines | InfoKids | 58 % | 1997/8/6 AM 01:54:26 | OK | 958 | 1297.htm |
| 7 | Industrial Technical Research Institute(ITRI) | ITRI Mechanical Industrial Journal | 58 % | 1997/8/7 PM 11:12:05 | OK | 1084 | 1588.htm |
| 8 | Motorcycle/Autobike<br>Motor Magazine | Motorcycle Magazine | 58 % | 1997/9/21 PM 12:32:47 | Timeout | 0 | 5194.htm |
| 9 | Motorcycle/Autobike<br>Motor & Motorcycle Magazine | Car Appreciation Magazine | 58 % | 1997/8/7 AM 09:20:48 | OK | 528 | 5435.htm |
| 10 | Technical Journals | CAD & Automation Magazine | 58 % | 1997/8/6 AM 01:54:22 | OK | 758 | 4466.htm |
| 11 | Technical Journals | Electronic Technology Magazine | 58 % | 1997/9/21 AM 10:10:22 | File Not Found | 0 | 4469.htm |
| 12 | Technical Journals | Mechanical Technology Magazine | 58 % | 1997/9/21 AM 10:10:23 | File Not Found | 0 | 4470.htm |
| 13 | Databases Services<br>Computer Journals & Magazines | Sing-ping Publishing | 58 % | 1997/8/6 AM 11:51:47 | OK | 1111 | 6247.htm |
| 14 | Electronic Online Shopping<br>Computer Journals & Magazines | Computer Street Mail Order Magazine | 58 % | 1997/8/6 AM 09:10:38 | OK | 554 | 6437.htm |
| 15 | Computer Journals & Magazines | Computer CD-ROM Magazine | 58 % | 1997/9/21 PM 12:32:47 | Timeout | 0 | 7830.htm |

Next

[ 1] [ 2] [ 3] Search All Objects

Fig. 14. Object-Level Search (Query Result: Objects in designated Classes).

Fig. 15 summarizes the results of "Search All Objects," as used by conventional search engines. This example contains a total of 746 relevant objects. In comparison with the 8 relevant classes returned by a class-level search, visiting and locating information in 746 links is extremely inefficient.

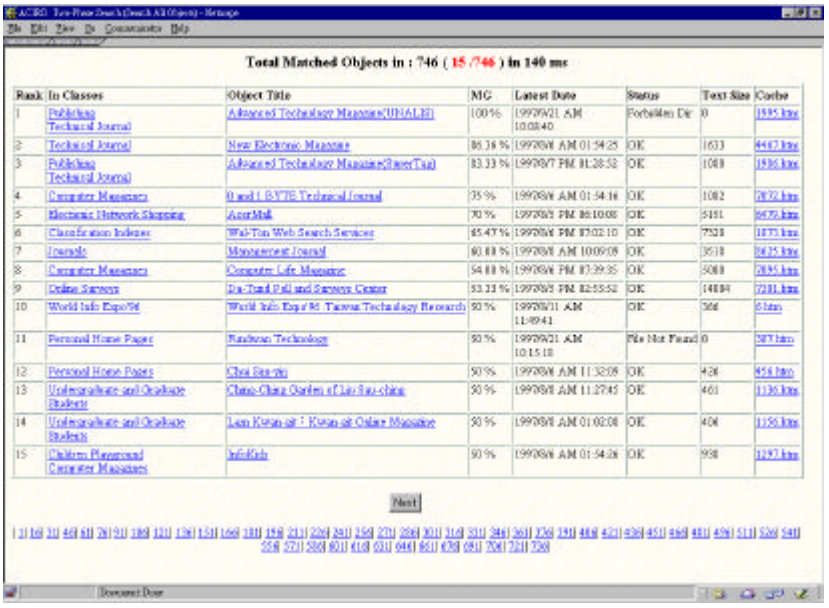



Fig. 15. Search All Objects (Query Result: All Objects).

## 8. Conclusions and Future Work

This paper has presented a class-based Internet document management and access system: ACIRD. Our results demonstrate that machine learning and data mining techniques are capable of generating accurate classification knowledge. Based on the classification knowledge, the classifier can automatically and satisfactorily classify Internet documents into classes in a class lattice. According to analysis of the user query log, the classification knowledge can serve as a *meta-index* to shrink the searching domain to enable retrieval of potentially desirable documents efficiently. In addition, class-level search returns with a comprehensible organization of classes. Users can associate their queries with the presented classes, navigate among the classes and finally perform object-level search to obtain their desired documents. In this manner, the system helps users discover information in a large number of Internet documents.

In the future, the learning model should be extended to incrementally learn the changes of the Internet. In addition, the classification accuracy of the learning methods and of the classifier still has room for further improvement. Some related issues are also worth closer examination and further study. For example, future work should examine the feasibility of extending the use of mining term associations in classes to automatically construct a thesaurus, which corresponds to the semantics of terms in the specific domain. Also, analyzing the user query log will allow the system to learn and extract new terms that cannot be found in thesauruses, such words like "MP3," "ICQ," and "CGI," in order to expand the term base of ACIRD.

## Acknowledgement

This work was partially supported by NSC under Contract NSC 88-2213-E-001-025.

## References

[1] 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.

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

[3] C. Apte, F. Damerau, and S. M. Weiss, "Automated Learning of Decision Rules for Text Categorization", ACM Transactions on Information Systems, Vol. 12, No. 3, July 1994, pp. 233-251.

[4] L. F. Chien, "PAT-Tree-Based Keyword Extraction for Chinese Information Retrieval", Proceedings of the ACM SIGIR International Conference on Information Retrieval, 1997.

[5] P. Clark and T. Niblett, "The CN2 Induction Algorithm", Machine Learning Journal, Vol. 3(4), 1989, pp. 261-283.

[6] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, "Introduction to Algorithms", MIT Press, 1990.

[7] W. B. Croft and P. Savino, "Implementing Ranking Strategies Using Text Signatures", ACM Transactions on Office Information Systems, Vol. 6, No. 1, Jan. 1998, pp. 42-62.

[8] D. Cutting, and J. Pedersen, "Optimizations for Dynamic Inverted Index Maintenance", the 13th International Conference on Research and Development in Information Retrieval, 1990.

[9] R. O. Duda and P. E. Hart, "Pattern Classification and Scene Analysis", John Wiley & Sons, New York, 1973.

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

[11] N. Fuhr, "Models for Retrieval with Probabilistic Indexing", Information Processing and Management, Vol. 25, No. 1, 1989, pp. 55-72.

[12] M. Goldszmidt and M. Sahami, "A Probabilistic Approach to Full-Text Document Clustering", TR ITAD-433-MS-98-044, SRI International, http://robotics.stanford.edu/users/sahami/papers-dir/gm-clustering.ps.

[13] Y. F. Jing and W. B. Croft, "An Association Thesaurus for Information Retrieval", UMass Technical Report 94-17, http://cobar.cs.umass.edu/info/psfiles/irpubs/jingcroftassocthes.ps.gz.

[14] K. S. Jones and D. M. Jackson, "The Use of Automatically-Obtained Classifications for Information Retrieval," Information Processing and Management (IP&M), Vol. 5, 1970, pp. 175-201.

[15] K. S. Jones and R. M. Needham, "Automatic Term Classification and Retrieval," Information Processing and Management, Vol. 4, No 1, 1968, pp. 91-100.

[16] T. Kalt and W. B. Croft, "A New Probabilistic Model of Text Classification and Retrieval", UMass Computer Science Technical Report, IR-78, 1996, http://cobar.cs.umass.edu/info/psfiles/irpubs/ir.html.

[17] L. S. Larkey and W.B. Croft, "Combining Classifiers in Text Categorization", ACM SIGIR' 96, 1996, pp. 289-297.

[18] T. Berners-Lee, "Hypertext Markup Language 2.0", http://andrew2.andrew.cmu.edu/rfc/rfc1866.html.

[19] D. Lewis, "An Evaluation of Phrasal and Clustered Representations on a Text Categorization Task", ACM SIGIR'92, pp. 37-50, 1992.

[20] D. Lewis and William Gale, "Training Text Classifiers by Uncertainty Sampling", ACM SIGIR'94, 1994.

[21] S. H. Lin, M. C. Chen, J. M. Ho, and Y. M. Huang, "The Design of an Automatic Classifier for Internet Resource Discovery", International Symposium on Multi-technology and Information Processing (ISMIP' 96), December 1996, pp. 181-188.

[22] 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.

[23] S. H. Lin, C. S. Shih, M. C. Chen, J. M. Ho, M. T. Kao, and Y. M. Huang, "A Collaborative Internet Documents Access Scheme Using ACIRD", International Computer Symposium on Software Engineering and Database Systems (ICS'98), 1998.

[24] R. S. Michalski, I. Mozetic, and J. Hong, "The AQ15 Inductive Learning System: An Overview and Experiments", Technical Report ISG 86-20, UIUCDCS-R-86-1260, Department of Computer Science, University of Illinois, Urbana, 1986.

[25] J. Mostafa, S. Mukhopadhyay, W. Lam, and M. Palakal, "A Multilevel Approach to Intelligent Information Filtering: Model, System, and Evaluation", ACM Transactions on Information Systems, Vol. 15, No. 4, October 1997, pp. 368-399.

[26] J. R. Quinlan, "Induction of Decision Trees", Machine Learning, Vol. 1, 1989, pp. 261-283.

[27] J. R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers. San Mateo, CA, 1993.

[28] G. Salton, "Automatic Information Organization and Retrieval," McGraw-Hill, 1968.

[29] G. Salton, C. Buckley, and C. T. Yu, "An Evaluation of Term Dependence Models in Information Retrieval," LNCS 146, 1983, pp. 151-173.

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

[31] G. Salton, "Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer", Addison Wesley, 1989.

[32] G. Salton and C. Buckley, "Improving Retrieval Performance by Relevance Feedback", Journal of American Society for Information Science, Vol. 41, No. 4, 1990, pp. 188-297.

[33] D. Shasha, and T. Wang, "New Techniques for Best-Match Retrieval", ACM Transactions on Office Information Systems, Vol. 8, No. 2, January 1990, pp. 140-158.

[34] 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.

[35] Y. Yang, "Expert Network: Effective and Efficient Learning from Human Decisions in Text Categorization and Retrieval", ACM SIGIR'94, 1994, pp. 13-22.

[36] C. T. Yu, W. Meng, and S. Park, "A Framework for Effective Retrieval," ACM Transactions on Database Systems, Vol. 14, No. 2, 1989, pp. 147-167.

[37] 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.

Shian-Hua Lin received his M.S. and Ph.D. degree in Engineering Science from National Cheng-Kung University, Tainan, Taiwan, in 1994, and 2000, respectively. He is currently a postdoctoral fellow of Institute of Information Science at Academia Sinica, Taipei. His research interests include Internet document classification and retrieval, machine learning, data mining, and database system.

Meng Chang Chen received the B.S. and M.S. degrees in Computer Science from National Chiao-Tung University, Taiwan, in 1979 and 1981, respectively, and the Ph.D. degree in Computer Science from the University of California, Los Angeles, in 1989. He joined AT&T Bell Labs in 1989 as Member of Technical Staff, and became an Associate Professor at Sun Yat-Sen University, Taiwan in 1992. Since 1993, he has been with Institute of Information Science, Academia Sinica, Taiwan. Currently, he is an Associate Research Fellow and Deputy Director. His current main research interests include database and knowledge base systems, knowledge discovery and representation, QoS networking, multimedia systems and streaming, and operating system.

Jan-Ming Ho received his Ph.D. degree in electrical engineering and computer science from Northwestern University in 1989. He received his B.S. in electrical engineering from National Cheng Kung University in 1978 and his M.S. at Institute of electronics of National Chiao Tung University in 1980. He joined Institute of Information Science, Academia Sinica, Taiwan, R.O.C as a associate research fellow in 1989,and is promoted to research fellow in 1994. He visited IBM T. J. Watson Research Center in summer 1987 and 1988, Leonardo Fibonacci Institute for the Foundations of Computer Science, Italy, in Summer 1992, and Dagstuhl-Seminar on "Combinatorial Methods for Integrated Circuit Design", IBFI-Geschäftsstelle, Schloβ Dagstuhl, Fachbereich Informatik, Bau 36, Universität des Saarlandes, Germany, in October 1993. His research interests target at the integration of theoretical and application-oriented research, including environment for management and presentation of digital archive, management, retrieval, and classification of web documents, continuous video streaming and distribution, video conferencing, real-time operating systems with applications to continuous media systems, computational geometry, combinatorial optimization, VLSI design algorithms, and implementation and testing of VLSI algorithms on real designs. He is

Associate Editor of IEEE Transaction on Multimedia. He was Program Chair of Symposium on Real-time Media Systems, Taipei, 1994 - 1998, General Co-Chair of International Symposium on Multi-Technology Information Processing, 1997 and will be General Co-Chair of IEEE RTAS 2001. He was also steering committee member of VLSI Design/CAD Symposium, and program committee member of several previous conferences including ICDCS 1999, and IEEE Workshop on Dependable and Real-Time E-Commerce Systems (DARE'98), etc.

Yueh-Min Huang was born in Taiwan, R.O.C., in 1960. He received the B.S. degree in engineering science from National Cheng-Kung University, Taiwan, R.O.C., in 1982, and both the M.S. and Ph.D degrees in electrical engineering from the University of Arizona, Tucson, AZ, in 1988 and 1991, respectively. Since 1991, he has been with the Department of Engineering Science, National Cheng-Kung University, where he is a professor. His research interests include distributed multimedia systems, machine learning, and real-time systems. Dr. Huang is a member of IEEE Computer Society, the Taiwaness Association for Artificial Intelligence, and the Chinese Fuzzy Systems Association. He was a winner of the 1996 and 1998 Acer Long-Term Award for Best M.S. Thesis Supervision.