ACIRD: Intelligent Internet Documents Organization and Retrieval

Shian-Hua Lin, Meng Chang Chen, Jan-Ming Ho, and Yueh-Ming Huang

Abstract

In this paper, we present an intelligent Internet information system ACIRD using machine learning techniques to organize and retrieve Internet Web documents. ACIRD consists of three parts: knowledge acquisition, document classifier and two-phase search engine. The knowledge acquisition of ACIRD automatically learns the classification knowledge from classified Internet Web documents and the classifier applies the classification knowledge to classify newly collected Internet Web documents to one or more classes in a class hierarchy. The experiments show that ACIRD performs as good as or better than human experts in both knowledge extraction and document classification. Based on the learned classification knowledge and the given class hierarchy, the ACIRD two-phase search engine presents hierarchically navigable structured results to the users instead of conventional flat ranked results that greatly helps users in discovering information from diversified Internet documents.

1. Introduction

The explosive growth of the Internet dramatically changes the way of working and living of all walks and the Internet becomes a major source of information and means of communication. However, the excessive information on the Internet creates the information overflow problem. To alleviate the problem, there are many Internet search engines available for the Internet users. Most of the search engines are implemented to facilitate rapid retrieval of documents for diverse users. Terms[1] are extracted from a document, stemmed, stored and indexed in database or other storage systems by applying indexing approach [12]. The user query is usually represented by a sequence of terms that are matched with the indexed terms based on TF×IDF algorithm or similar algorithm [9] to retrieve relevant documents. In order to distinguish the relevance of the documents to the query, the retrieved documents are presented in a ranked list.

[1] In the paper, "term" is considered as the "word" extracted from a document. In this paper, "keyword" is representative of a concept, while "term" may not be meaningful.

From the perspective of retrieval efficiency and effectiveness, word-based information retrieval (IR) systems are efficient in handling a large document base. However, documents collected from the Internet are extremely numerous. In such case, for a query with two words[2] submitted to a search engine implemented with similarity-based algorithms [8, 9], thousands of documents are probably retrieved. For the query example “education and university”, there are 87,368,493 hits by AltaVista, 7,379,086 hits by Infoseek, 237,902 hits by WebCrawler and 2,879 hits by Yahoo. Ranking a large number of documents using very few words is not likely to produce an order of documents meeting the preference of the user. Consequently, user has to retrieve many uninteresting documents before obtaining the desired information. Several search engines has applied relevance feedback [34] to expand or refine the query based on documents selected by the user. However, query expansion based on the selected documents may not be effective since the user intention is difficult to grasp from the feedback.

The conception gap between the web document developers and the user creates the difference between the retrieved results and the user expectation. Due to the richness of language and culture, web developers and users may use different terms to represent the same concept, or use the same term to describe different things. Therefore, word-based search engines often retrieve documents, probably not desired by users. As a result, Internet search engines generally retrieve thousands documents with few desired. In contrast, desired documents may not be retrieved. For instance, the term “airline schedule” in documents does not match the term “flight schedule” in the query by a word-based search engine, but both terms are considered to have the same meaning. However, a thesaurus for the whole domain still cannot solve the problem, since a word may have different meaning in different contexts, such as the word “bank”. A thesaurus for each specific domain can help to alleviate the problem. But, due to the diversity and dynamic nature of the Internet, no static thesaurus can handle the shifting semantics of terms.

Ideally, Internet search engines should be able to retrieve relevant documents efficiently, and present the documents in accordance with user expectation. For the efficient retrieval, the search engine should be able to shrink the document search space, since it is also possible to have more documents than the physical memory can store. As for effective retrieval, the system needs to draw close the semantics mismatch between queries and documents. Assigning classes to documents is essential to the efficient management and retrieval of knowledge [10], and also provides a

[2] According to the statistics in [13], the average query length is 1.3 words.

framework for structured query processing. Our system, *ACIRD*[3] *(*Automatic Classifier for the Internet Resource Discovery) [7, 11] is designed to automatically classify documents into proper classes in a class hierarchy provided by Yam[4]. The system is capable of learning classification knowledge from classified documents and mining the association rules among terms to explore the implicit term semantics. ACIRD also infers from the term associations to refine the classification knowledge of each class. To reply user query, the system implements a two-phase search that presents a hierarchically navigable view, based on the discovered classification knowledge and the given class hierarchy, to the user.

In the rest of the paper, we discuss related work in Section 2. The system, ACIRD, is described in Section 3. In Section 4, we define the terminology used in this paper. In order to illustrate the system clearly, in Section 5, we describe the learning model of ACIRD. Section 6 shows the experiments of automatic classification to justify to the design and implementation of ACIRD. Then the two-phase search approach is introduced in Section 7. Finally, we conclude the contribution and point out future work.

2. Related Work

Past studies in Information Retrieval (IR) systems and search engines focus on the improvement of retrieval efficiency by using indexing and query reformulation techniques. The first step of word-based document processing is to extract words from documents based on pre-constructed dictionary, stoplist, and stemming rules. Once words are extracted, a widely used method TF×IDF [9, 23] is applied to determine the weights of words. Term frequency (TF) is the number of occurrence of a word in a document and inverse document frequency (IDF) is the inverse of the document frequency, defined as the number of documents in which the word occurs. The weight of a word can be determined by the product of its TF and IDF for the TF×IDF method. A document thus can be represented by a set of words and their weights, called *vector of weighted words* representation. The similarity function of two documents or a query and a document is the direct product of their vectors of weighted words, the cosine value between the two vectors in a multi-dimensional vector space.

Another popular approach, string-based indexing, indexes strings and all their sub-strings, instead of words, in the document. The string-based indexing approach is particularly useful for the

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

[4] http://www.yam.org.tw/b5/yam, which is a popular search engine in Taiwan.

applications of full-text search, exact string match (e.g. address) and character-based language (e.g. many oriental languages) that search for arbitrary-length strings. In comparison with word-based indexing approaches, the storage requirement of string-based indexing approach patterns is much larger. In addition, their complicated data structures take more time in retrieval. As retrieving exact matched strings only, string-based indexing approaches do not fit for many Internet information discovery queries that only give conceptual descriptions instead of exact strings. Some string-based indexing technologies, such as PAT-tree [25], are proposed to improve the performance of various search functions, such as prefix searching, proximity searching, range searching, longest repetition searching, most significant and most frequent search, and regular expression searching [12]. However, these searching functions are rarely used in the Internet document search.

Both of word-based and string-based indexing approaches require large storage space for maintaining the indexes, even larger than original documents. For a huge document base such as the Internet, the physical memory up to many giga bytes is required to retain good performance. Otherwise, the page faults of operating system will dramatically degrade the performance. Generally it is a good idea to organize the indexes in a hierarchical structure in order to reduce the access time and memory requirement. To further reduce access time and memory, the organization of hierarchy has to fit the user retrieval patterns. In this paper, we propose a hierarchical indexing approach that each internal node of the hierarchy represents a class, as the concept of class fits well with many Internet information discovery procedures. Using machine learning techniques, our system is capable of learning classification knowledge from training documents. The classification knowledge is the first-level index. Combining with the second-level index (e.g. keyword-index and string-index), our system provides a hierarchical index structure for efficient document retrieval.

There are several approaches to the classification of documents, including manual classification, automatic classification that can be further divided into classification knowledge acquisition from domain experts and automatic learning of classification knowledge. Manually assigning classes to documents is time consuming and expensive. Knowledge acquired from domain experts, while is relatively effective, is also expensive in time and efforts of development and support. Furthermore, the acquired knowledge may be incomplete. Classification knowledge automatically learned from training document is efficient in time and cost, but its accuracy is limited by the employed learning model.

There are many *text categorization* studies in the information retrieval discipline [4, 10, 20, 21, 22, 31]. In this paper, we use “document classification” instead of “text categorization”, since we focus on the Internet HTML documents rather than general texts. Document classification is the problem of automatic documents grouping. Many studies also deal with the problem of document retrieval, relevance feedback, text categorization, routing, filtering, and clustering. For example, ExpNet [35] uses similarity measurement as the category ranking method to determine the best matched category as the classification of the input document. [needs more examples]

Past studies in machine learning developed many algorithms that have been well tested and performed in many domains such as medical, finance, etc. To name a few popular algorithms, there are ID3 [27], C4.5 [28], CN2 [29], and AQ algorithm [30]. However, these algorithms are applied to structural training data instead of non-structured textual data. This motivated many approaches to document classification use corpus to characterize documents and develops new algorithms to learn classification knowledge. These algorithms include Bayesian independence classifier [21], k-nearest-neighbor [22, 32], rule-based induction algorithm [10], and mixed approached such as INQUERY [33]. Those systems concentrate on the document categorization and the learning algorithm, but they omit the diversity of the semantics of terms (or features) in the document. In machine learning, the feature is usually an attribute-value pair that its semantics is certain. However, the semantics of document feature is uncertain, and varies with different domains. For example, the document feature “apple” has different meanings for the domain “computer” and “food”.

In this paper, we apply the technique of mining association rules to explore the semantics of document feature in different domains. Mining association rules [16, 17, 18] is applied to discover the important associations among items in transactions. A well-known application of mining item associations is to discover an optimal item arrangement in the supermarket to allow customers to gather their grocery conveniently.

3. The ACIRD System

One of the main goals of the system, Automatic Classifier for Internet Resource Discovery (*ACIRD*) [11], is to automatically collect and classify Internet documents for efficient and effective management and retrieval. The initial motivation of ACIRD is to improve the expensive and time-consuming manual classification process used by many Internet search engines. Employing the classification knowledge learned from the manually classified Internet documents, the system automatically classifies incoming Internet documents. The classification knowledge together with the class hierarchy enables two-phase search in the document retrieval. This section gives the overview of ACIRD, and the details of the system are presented in the following sections.

The knowledge base of ACIRD is learned from the Internet documents collected and manually classified in Yam that the Internet documents are assigned to one or more classes in a class hierarchy (or more precisely, a class lattice). An Internet robot is implemented to automatically collect HTML documents. The system assigns a unique object ID to every document (called *object* in this paper) and stores the object in the database. Each HTML document is parsed into a set of terms with weights calculated from term frequency and weight of HTML tags. The terms and weight pairs form the feature vector to represent the object knowledge. Inverted index of terms to their objects is implemented using commercial relation database management system for convenient access during learning and searching.

The classification learning process of ACIRD consists of two phases: training phase and testing phase. In the training phase, given a class hierarchy, a collection of manually classified documents is the training data. The learning process starts from the most specific classes to the most generalized classes of the class hierarchy. For the most specific classes (i.e. the leaves of the class hierarchy), the class knowledge is generalized from the knowledge of objects in the class. For other classes, the knowledge is generalized from its child classes. In the testing phase, a classifier is built by incorporating the learned class knowledge. The newly collected documents manually classified by experts are compared with the class assignment of the classifier in order to verify the quality of learned knowledge.

After classification learning process, the technique of mining association rules is employed to mine term associations to enhance the class knowledge. As term associations highly depend on the class domain, the mined term associations can only be applied to refine the classification knowledge of the specific class. We showed that the mined term association is effective to enhance the term semantics from the experiments in [7].

ACIRD has a two-phase search engine that allows users to retrieve interesting documents, represented as a class hierarchy rather than a sequence of ranked documents, effectively and efficiently. In the two-phase search, a query string is parsed and formulated as a sequence of terms, called the query feature vector. Similarity match based on Vector Space Model is applied to decide the relevance between the query and the classes and objects, which are also represented as feature vectors. In the first phase, *class-level* search is performed that the query feature vector is used to match qualified classes. Those qualified classes form a shrunk view of the class hierarchy. If user decides to further explore a qualified class in the hierarchy, the query feature vector is again employed to match subclasses of the class in the second phase. In *object-level* search, ACIRD matches and retrieves and qualified documents in the class. The two-phase search approach not only reduces the search domain, but also provides a hierarchical conceptual view that is effective to help the user to discover the information.

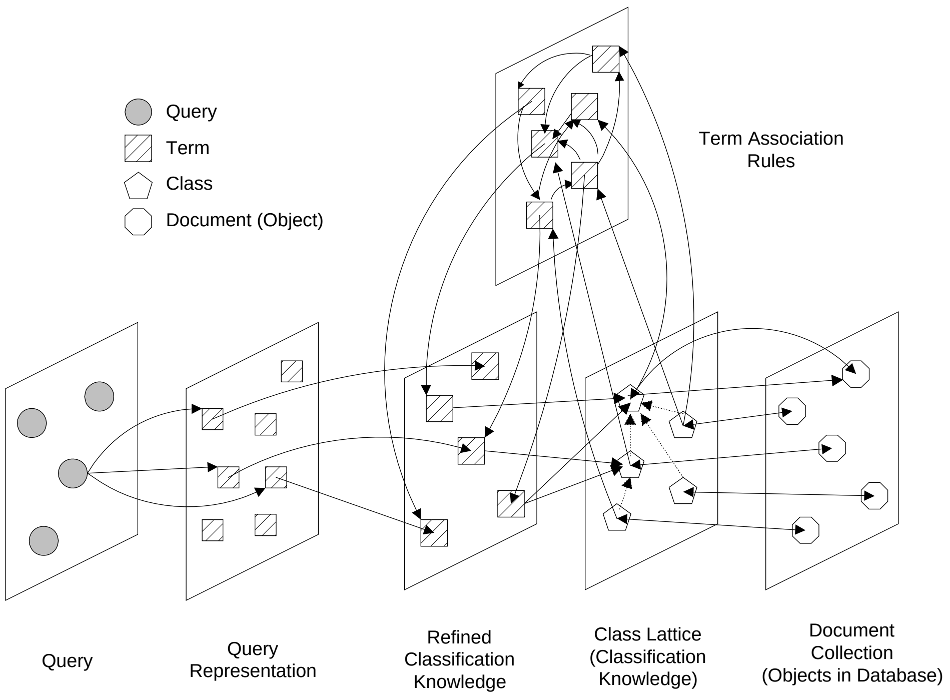

Query
Term
Class
Document (Object)
Term Association Rules
Query
Query Representation
Refined Classification Knowledge
Class Lattice (Classification Knowledge)
Document Collection (Objects in Database)

Fig. 1 The overview of the ACIRD

In Fig. 1, we summarize the system. First, *Document Collection* and *Class Lattice* are originated from Yam, and every document in the document collection is assigned one or more class in the class lattice. From right to left of Fig. 1, terms are extracted from documents. By applying learning algorithms, the knowledge of each class, *Classification Knowledge*, is obtained. By mining *Term Association Rules* of every class, the classification knowledge is refined to *Refined Classification Knowledge*, which is used to construct the automatic classifier and for the class-level search. From left-hand side, two-phase search engine parses the submitted *Query* into *Query Representation* that is used to match *Refined Classification Knowledge* to discover relevant classes (concepts) or to match and retrieve the interesting documents.

4. Notations and Definitions

In this section, we define the terminology used in the paper and introduce the conceptual model and knowledge representation of ACIRD. We denote an entity as a lower case letter, and a set or series of entities as an upper case letter. For example, let c denote a class and C denote a set of classes. In the following, we describe the system entities with their notations in parentheses from the higher level concept to the lower level concept.

- *ACIRD* learning system automatically learns and refines classification knowledge from a given class hirarchy and a set of manually classified objects. The hierarchy is called *ACIRD Lattice* because of the the hierarchy satisfies the lattice properties.
- *ACIRD Lattice* ($L_{ACIRD}(C,R)$) is set of classes C and a set of relations R between classes that can be represented as a graph with nodes (C) and edges (R).
- *Class* (c) is a class node of L_{ACIRD}. c possesses the knowledge generalized from the sub-classes or objects in the class.
- *Object* (o) is an HTML document that consists of paragraphs (pg) enclosed by HTML tags. An object o can belong to one or several classes in L_{ACIRD}.
- *Paragraph* (pg) consists of a series of sentences (s) that in term contain phrases and terms. pg is informative if it is enclosed by informative HTML tags that are defined later in this paper. Separators such as comma, period, semicolon, or other characters specified in the system are used to identify the boundary of *sentences* (s).
- A *phrase* is a sequence of terms in a sentence that are frequently used in some class domain and satisfies one of the specified phrase rules. To automatically construct thesauras, *Phrase Discovering Process* applies *phrase rules* to find new phrases satisfying the rules. For example, the simple noun phrase rule {NNN, NN} means that a phrase can be triple co-occurrence nouns or double co-occurrence nouns. Currently, the construction of thesauras of ACIRD is a semi-automatic process that the system extracts discovered candidates for the selection of users.
- *Term* (t) is a word (excluding stop words) extracted from sentences of an informative prargraph. Each term has a support value to the object that it appears. The *support* ($sup_{t,o}$) of t to o is calculated from 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 to the object. $Know_o$ can be represented by the *Term Support Graph* ($TSG(T,o,E)$) that 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 term *T* that each term t_i has a support value $sup_{t,c}$ to c. $Know_c$ is generalized from $Know_o$ or classification knowledge of its child classes. Similar to $Know_o$, $Know_c$ can be represented as a graph $TSG(T,c,E)$ that each directed edges in *E* from t_i (in *T*) to *c* is labeled with $sup_{t_i,c}$. 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*. For each pair of terms, t_i and t_j, there is a corresponding *confidence* ($conf_{t_i,t_j}$). 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 with $conf_{t_i,t_j}$.

– For each class *c*, *Term Semantics Network* ($TSN(T,c,E)$) is constructed as the union of the $TSG(T,c,E)$ and $TAG(T,E)$. TSN is used to represent the semantics of the class and the associations between terms in the class.

– *Perfect Term Support* (*PTS*) algorithm [7] is applied to promote $sup_{t_i,c}$ of edges in $TSN(T,c,E)$. The algorithm finds an *optimal path* ($p^*_{t,c}$) from t to c, where $p^*_{t,c}$ is a path with the maximal value among all the possible paths ($P_{t,c}$) from t to c in $TSN(T,c,E)$. The value of $p_{t,c}$ is the product of edges' confidence values in the path and the support to *c* of the last term, ($conf_{t,t_j} \times conf_{t_j,t_k} \times ... \times conf_{t_y,t_z} \times sup_{t_z,c}$). 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* that filters out terms whose $sup^*_{t,c}$ are less than the specified threshold θ_C. For a keyword, its $sup^*_{t,c}$ is defined as *membership grade* ($MG_{t,c}$) of t to c. The application of PTS and Filtering Process refine $Know_c$ to become *Refined Classification Knowledge* ($Know^*_c$). $Know^*_c$ is the knowledge base employed by *Two-Phase Search Engine* and *Automatic Classifier* of ACIRD.

In Fig. 2, the top-down view of these abstractions is shown in the left-hand-side and the systematic knowledge representation is shown in the right-hand-side.

錯誤! 連結無效。

Fig. 2 Conceptual model and systematic knowledge representation of ACIRD.

5. The Learning Model

In this section, we describe the learning model of ACIRD in details. In the training phase, ACIRD adopts supervised learning techniques and regards previously classified documents as the training objects. The testing phase is described in Section 6.

ACIRD applies machine learning methods to learn classification knowledge as shown in Fig. 3. The learning model is applied to each class of ACIRD lattice from the most specific classes to the most general class. Each document is preprocessed into a weighted term vector. Then the dimension of the vector is reduced by Feature Selection Process to reduce the complexity of learning and noise. For the most specific class, the knowledge of the class $Know_c$ is generalized from the knowledge of all objects in the class, which can be represented by Term Support Graph (TSG). Mining association algorithm is applied to mine associations of terms in TSG that the obtained term associations can be represented by Term Association Graph (TAG). By combining TSG and TAG, Term Semantic Network (TSN) is derived. TSN is further optimized to become TSN* to represent $Know_c^*$ of the class. For each iteration, only one term is promoted. As the promotion may be used to promote other terms, the promotion process is applied recursively until the stable state is reached. For the non-most specific classes, the learning process is the same except the initial weighted term vector is from its subclasses and objects.

錯誤! 連結無效。

Fig. 3 The Learning Model of ACIRD.

5.1 Preprocessing Process and Knowledge Representation

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

HTML Parser

An HTML document consists of paragraphs that the associated HTML tags indicate their importance and meta-level information. Web developers highlight the contents by HTML tags [reference], such as TITLE, Hn (headings), B, I, U, etc. In addition, META tag allows developers to add extra information to the document such as “CLASSIFICATIONS” and “KEYWORDS”. Apparently, the implication of tags is necessary to be considered during indexing the documents. Currently, human experts assign weights to HTML tags from observing the outcomes of numerous experiments. We classify these tags into four types.

- *Informative*. The paragraph enclosed by such tags are the the meta knowledge of the documents or the contents presented to the users. Thus, the tags have higher weights than the others. For example, CLASSIFICATION and KEYWORD in META, TITLE, Hn, B, I, U, etc.
- *Skippable*. Tags, such as BR and P have no effects in the semantics of the document and are omitted.
- *Uninformative*. Contents enclosed by tags, such as AREA, COL, SCRIPT, COMMENT, etc., are invisible from the users. Thus, these tags and their contents are excluded.
- *Statistical*. Contents included in these tags, such as !DOCTPYE, APPLET, OBJECT, SCRIPT, etc., are stored in database for statistics purpose.

The implementation of HTML Parser uses two stacks that one is for HTML tags and the other is for paragraphs. The execution of the algorithm is completed in a file scan.

Term Parser

Term Parser partitions the paragraph into sentences, extracts terms in the sentence, and counts the term frequency (TF) of each term. As designed to handle multi-lingual documents, ACIRD currently considers English and Chinese languages. For English, each extracted term is stemmed. For the character-based language, like Chinese, a sentence needs to be segmented into meaningful multi-character terms. As there are no apparent stop characters, Term Parser uses a pre-constructed term base of multi-character Chinese terms to extract meaningful terms. 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). The value, normalized to be in the range of [0, 1], indicates the importance of a term to represent the object.

$$\begin{aligned}
&sup'_{t_i,o} = \sum_{T_j} tf_{ij} \cdot w_{T_j}, \text{ where } t_i \text{ is a term in the sentence described by tags } T_j, \\
&\qquad\qquad tf_{ij} \text{ is the term frequency of } t_i \text{ in the sentence described by } T_j, \text{ and} \\
&\qquad\qquad w_{T_j} \text{ is the maximum weigh of tags in } T_j. \qquad (5.1) \\
&sup_{t_i,o} = \frac{sup'_{t_i,o}}{\underset{t_i \text{ in } o}{MAX}(sup'_{t_i,o})}, \text{ i.e., } sup \text{ is normalized to } [0,1]
\end{aligned}$$

Since a sentence may have more than one tag, the tag with the maximum weight is used in calculating the term support. We use TF and the maximum tag weight to calculate the term support rather than the TF×IDF weighting approach. Inverted Document Frequency (IDF) is designed to enhance the discriminating capability of high-frequency terms, which is not critical in supervised learning, as in ACIRD.

Term Parser extracts Chinese terms based on the heuristics of "the longer term first" to resolve the ambiguity. I.e. for two terms that one term is part of the other term, Term Parser will choose the longer one. A pre-constructed term base is built as a B*tree [36] for linear time access. In addition, the rules for Chinese term segmentation are supplied to handle the ambiguity of segmentations 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 HTML Parser, the complexity of Preprocessing Process is $O(N^2)$, where *N* is the content length of an object.

5.2 Feature Selection Process

After HTML Parser and Term Parser process an object, the obtained object knowledge can be 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, induction process can be applied immediately to learn the classifcation knowledge from the object knowledge. In practice, the complexity of the learning process is exponentially increased by the dimension of the vector and the noise may be increased with the increase of the dimension. Feature Selection Process is designed to reduce the complexity and noise during learning process. For each object, a pre-defined threshold of support θ_s is used to discard less important features, and remaining features are used to represent the object knowledge $Know_o$. In this way, the problem of feature selection is shifted to the selection of θ_s. Higher θ_s discards more features that the remaining terms may not be sufficient to represent $Know_o$. In contrast, lower θ_s has little effect in feature selection. In ACIRD, the selection of θ_s is adaptive to the emperical experiments. For instance, by analyzing the distribution of term supports from the training data as shown in Fig. 4, we observe that more than one half of term supports are in the range [0, 0.2). If we choose $\theta_s = 0.2$ to filter out terms with low supports, the average number of terms in an object is reduced from 28.64 to 11.61. It is obvious that the complexity of the feature selection is linear to the number of features.

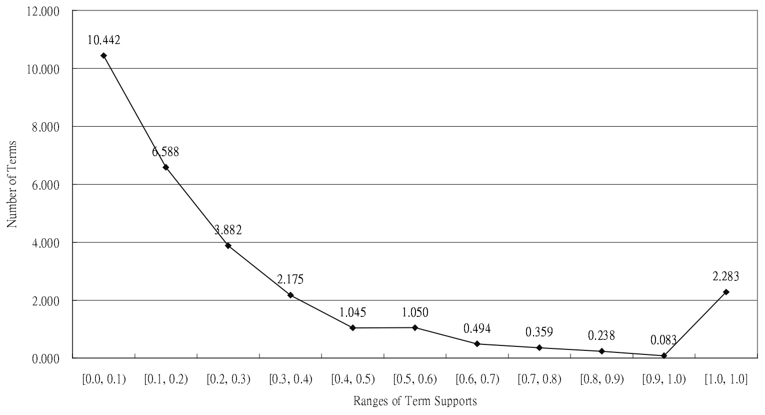


Fig. 4. The distribution of term supports of training data.

5.3 Classification Knowledge Learning

To discover the classification knowledge, induction learning is used to generalize the object knowledge $Know_o$ to the most specific class knowledge $Know_c$, and further generalize the class knowledge to its supper classes. The induction process is applied from the most specific to the most general classes. The class assignments of training objects are given by the human experts of Yam. In the conventional learning schemes, the values of features of training objects are either TRUE or FALSE. I.e. the learning algorithms generalize the term t_i to the class c based on the occurrence of objects containing t_i in *c*. In other words, it considers all the terms are equally important that ignores the degrees of term supports to the object or class. To amend the shortfall, 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].

$$sup'_{t_i,c} = \sum_{o_j} sup_{t_i,o_j},\ sup_{t_i,o_j} \text{ is the term support of } t_i \text{ to } o_j,\ o_j \text{ is an object in the class } c.$$

$$sup_{t_i,c} = \frac{sup'_{t_i,c}}{MAX\{sup'_{t_i,c}\}}, \text{ i.e., } sup'_{t_i,c} \text{ is normalized to } sup_{t_i,c}\text{, which is ranged in } [0,1] \qquad (5.2)$$

Similar to Eq. (5.2), the support of term to non-most specific class can be obtained from the support of term to its objects and subordinate classes, as shown in Eq. (5.3). The number of objects in a subordinate class affects its contribution to the super class.

$sup'_{t_i,c} = \sum_{o_j} sup_{t_i,o_j} + \sum_{c_j} \|c_j\| \times sup_{t_i,c_j}$, sup_{t_i,c_j} is the term support of t_i to c_j,

c_j is a subordinate class of class c, and $\|c_j\|$ is the number of objects in c_j. (5.3)

$sup_{t_i,c} = \frac{sup'_{t_i,c}}{MAX\{sup'_{t_i,c}\}}$, i.e., $sup'_{t_i,c}$ is normalized to $sup_{t_i,c}$ in $[0,1]$

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

1. For each class from the most specific to the most general, do the preprocessing and feature selection processes.

2. For most specific class, calculate the term supports to class based on Eq. (5.2). The complexity is the sorting complexity, $O(N_o + NT_o \log NT_o)$, where N_o is the number of object in the class, and NT_o is the maximum number of terms in an object of the class.

3. If the class has subordinate classes, calculate the term supports to class based on Eq. (5.3). The complexity is $O(N_o + N_c * NT_c + NT_c \log NT_c)$, where N_c is the number of subclasses, and NT_o is the maximum number of terms in a subordinate class.

As the number of term in a class is large and due to the diversity of Inteernet documents, the term supports to the class is generally low. For instance, from the statistics of training data, there are about 472 terms per class in average and the supports are low as shown in Fig. 5. From the figure, most term supports locate in 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 θ_c=0.1, in average there are 47 terms per class, 24 terms with $\theta_c = 0.2$, and 20 terms with $\theta_c = 0.3$. A filtering process may remove meaningful but with low support terms, which include aliases of high support terms and terms highly associated with them. To alleviate the problem, we propose a mining association rules approach to discover term associations in a class, and apply the result to enhance the supports of the otherwise filtered out terms.

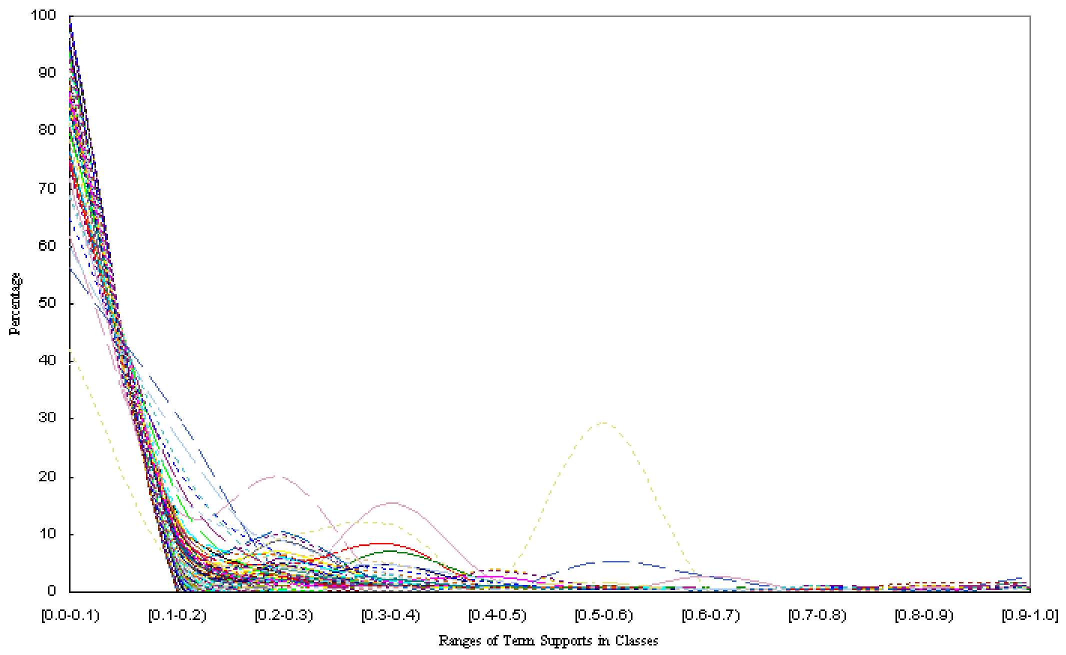


Fig. 5. The distribution of term supports of all the classes in ACIRD.

5.4 Mining Term Association

The feature selection process in the class level is more complicated than in the object level. The first reason is, in general, $\|Know_c\|$ is larger than $\|Know_o\|$, since $Know_c$ is generalized from several $Know_o$. The second reason is that the terms in an object are more consistent in both semantics and representation than in a class. Usually an object is written by one web developer that a simple filtering method using a threshold value can do a reasonably good job, while the objects in a class are collected from many web servers and written by a diverse of web developers that add diversity in the term wording and usage. Often applying a filtering algorithm using a threshold value θ_c directly to $Know_c$ removes many representative terms but with low support values. As only few concepts remain, the recall rate on $Know_c$ is likely low. Therefore, the system must identify and consolidate terms with or related to the same concept before applying filtering process. In ACIRD, we propose mining term associations and perfect term support algorithm to promote representative terms with low supports.

According to the definition of association rule in [17], elements in the problem are *items*, *transactions*, and the *database*.

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items and D be a set of transactions (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 $X \cap Y = \phi$. 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 that contain $X \cup Y$.

We follow the above definition and map the problem of mining term associations to the specification of mining association rules. Two important issues should be considered before mining association rules [16, 17, 18]. One is the granularity (or the transaction defined in [17]) used to mine associations. The other is the domain used to generate association rules, which is corresponding to the transaction database defined in [17].

Granularity of mining associations.

In [19], the system restricts the granularity of generating associations to 3-10 sentences per paragraph in order to reduce the computational complexity. The restriction is impractical for web documents since a paragraph may have hundreds of meaningful sentences. Additionally, the importance of a sentence in web document depends on the HTML tags, not its position. Therefore, the granularity of mining term association is the whole informative paragraphs.

Domain of generating association rules.

As Internet documents are published by diverse people, the semantics of a term depends on the authors and contexts that it is common to have the same word to represent different meanings. For example, if a document mentions “apple computer” in a paragraph, the semantics of the apple should not be “apples of fruit or food”. Most likely, it indicates “Macintosh” in the class of “Computer”. Similarly, “apple” and “pie” means the apple of fruit in the class of “Food”. The above observation supports the idea to restrict 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 makes their associations good candidates for mining. Based on these reasons, ACIRD applies the mining association rules process [17] to mine term associations by regarding

(i) *Terms* are corresponding to *items*.

(ii) The object’s *informative paragraph* is corresponding to the *transaction*.

(iii) The *class* is corresponding to the *transaction database*.

Concentrating on objects of a class instead of all objects also has the advantage of small database size, as the complexity of mining associations is exponentially increased with the size of the database. When the size of database is not large, a simple mining association algorithm, such as Apriori [16], can be efficiently applied.

For the definitions of *confidence* and *support* [17] of term association $t_i \rightarrow t_j$, we do the following modifications.

$conf_{t_i -> t_j} = \frac{df_c(t_i \cap t_j)}{df_c(t_i)}$, where $df_c(t_i)$ stands for the number of documents that contain term t_i.

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

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

After mining term associations of a class, TSN is obtained, as shown in Fig. 6. TSG represents the term supports of a class, TAG represents the term associations in a class, and TSN is the union of TSG and TAG, i.e., $TSN(T,c,E) = TSG(T,c,E) \cup TAG(T,E)$.

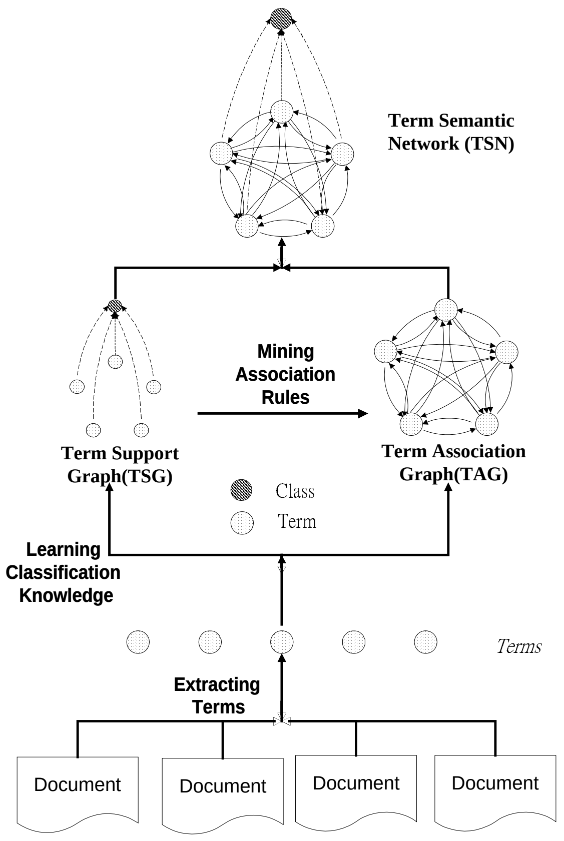


Fig. 6. Construction of Term Semantic Network

5.5 Classification Knowledge Refinement

As the term associations are asymmetric, both TAG and TSN form strongly connected digraphs. In order to decide sup^*, all the possible paths from the term to the class need to be considered. For a TSN, the number[5] of all the possible paths from terms to the class is $n \cdot \sum_{i=1}^{n-1} P_i^{n-1}$. In ACIRD, the average number of terms of a class is 472 that exhaustive search is infeasible. Although the support value of term association can be employed as a filter to remove rarely used term, it is still computationally expensive for a small number of terms. For instance, a class with 10 terms creates 2.3×10^8 possible paths. Therefore, an efficient algorithm is required.

Here we propose the PTS algorithm to find $sup^*_{t_i,c}$'s for all terms in a class in polynomial time. From the definition $sup^*_{t,c} = MAX\{conf_{t \to t_j} \times conf_{t_j \to t_k} \times ... \times conf_{t_y \to t_z} \times sup_{t_z \to c}\}$, and $conf$ and sup range in [0, 1], the more edges are involved in the path $p(t, t_j, t_k, ..., t_y, t_z, c)$, the smaller value

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

their product is. In other words, a sub-path of an optimal path must be an optimal path too. The proposed greedy heuristic is as follows.

Heuristic. Divide the nodes in TSN into two groups T and T^*. T contains all term nodes and T^* is empty in the initial state. Each time find a node t from T has the maximum $sup^*_{t,c}$, t is moved from T to T^*. The heuristic is recursively applied until T is empty.

The PTS algorithm is shown below.

Perfect Term Support (PTS) Algorithm

```
1. [Initial state: This step initializes every $sup^*_{t_j,c}$ and partitions the terms into two groups: $T^*$
   contains the terms that have been optimized and $T$ contains terms that are not optimized.]
   Let $sup^*_{t_i,c} \leftarrow sup_{t_j,c}, \forall t_j \in c$;
   Let $sup^*_{t_i,c} \leftarrow MAX\ \{sup^*_{t_i,c} \mid \forall \mathrm{t_i} \in \mathrm{c}\}$;
   Let $T^* \leftarrow \{(t_i, sup^*_{t_i,c})\}$; $T \leftarrow c - \{t_i\}$;
2. [This step updates $sup^*_{t_j,c}$ for each term in $T$, if necessary. $t_{last}$ indicates the latest term
   added into $T^*$.]
   If $T$ is not empty, continue Step 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 finds the term with maximal $sup^*_{t,c}$ from $T$ and inserts it to $T^*$.]
   Let $t_{last} \leftarrow \{t_k \mid t_k \in T$, and $sup^*_{t_k,c} = MAX\{sup^*_{t_j,c} \mid t_j \in T\}\}$;
   $T \leftarrow T - t_{last}$; $T^* \leftarrow T^* + \mathrm{t_{last}}$; Output $(t_{last}, sup^*_{t_{last},c})$.
```

In [7], we have proved that the PTS algorithm always finds the optimal solution in time complexity $O(\|Know_c\|^2)$. PTS can efficiently promote some *non-representative* terms by exploring their associations with *representative* terms. Fig. 7. illustrates the effect of PTS. In the left-hand side, there are four *non-representative* terms in TSN. After refinement using PTS, in the right-hand side of the graph, three terms are promoted to the *representative* for their associations to the *representative* term. All other non-representative terms and associations are eliminated to reduce the complexity of learning.

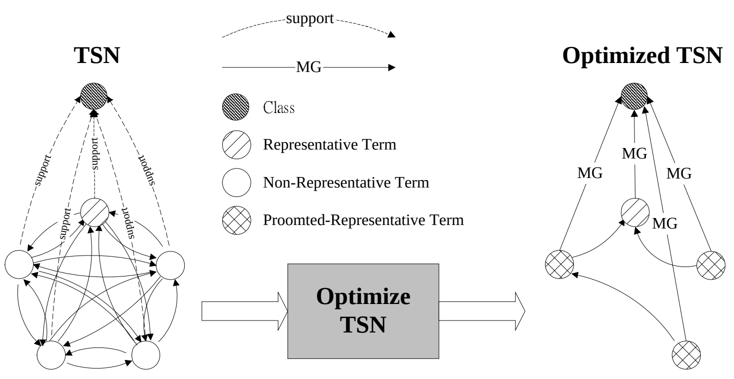


Fig. 7. PTS Refinement on TSN

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 $Know_c$ and $Know_c^*$ with the keywords selected by ten human experts. The experiment shows Knowledge Refining Process indeed refines the knowledge contents of $Know_c$, and it also demonstrates the trade-off between precision and recall based on different criteria of threshold. There are two types of criteria used to evaluate the outcomes of the experiment.

- **Top n**. All the $sup_{t,c}^*$ are sorted in descendent order. The first *n* terms are selected to be the keywords of $Know_c^*$.
- **Threshold = ?** . This criterion selects terms with $sup_{t,c}^* \geq \theta$.

The experiment results are shown in Table 1. Before applying PTS, the lowest precision is 0.76, due to the high selection standards (Top 10, Top 20, T = 0.5, and T = 0.7) that select highly informative terms only. However, the recall is low for the same reason. It implies that the Induction Process does not learn the implicit association among terms, although it generalizes the terms of objects to class. In comparison with the case without applying PTS, PTS increases both precision and recall for the **Top n** criterion as it promotes important but non-representative terms at the cost of removing less important terms (as **Top n** criterion selects a fixed number of terms). For the "**Threshold = ?**" criterion, PTS increases recall while decreases precision since it promotes terms to the keywords of the class. Hence, by applying Induction Process and Knowledge Refining Process, the hidden semantics among terms can be discovered. With the carefully chosen selection criterion, the acceptable compromise of precision and recall can be achieved.

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

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 similarity concept, we implement an automatic classifier, *ACIRD Classifier*, to clssify newly collected Internet objects. For each object, the classifier assign one or more classes which, in the testing phase, are compared with the classes assigned by human experts to evaluate the classification accuracy. Based on a series of experiments and analyses, it can be observed that $Know_c^*$ is effective to support automatic classification of the Internet documents.

Similarity Measurement

ACIRD Classifier uses the conventional similarity measurement, the cosine value of feature vectors of document and class, defined in equation (6.1).

$$sim(o,c) = \frac{\sum_{t_i \text{ in } c \text{ and } o} (sup_{t_i,o} * mg_{t_i,c})}{\|o\|_2 \times \|c\|_2}, \text{ where } t_i \text{ is a common term of } o \text{ and } c,\ sup_{t_i,o} \text{ is the support}$$
$$\text{of } t_i \text{ to } o, \text{ and } mg_{t_i,c} \text{ is membership grade of } t_i \text{ to } c, \text{ i.e., } sup^*_{t_i,c}; \quad (6.1)$$
$$\|o\|_2 = \sqrt{sup^2_{t_1,o} + sup^2_{t_2,o} + \ldots} \text{ is the norm of the object;}$$
$$\|c\|_2 = \sqrt{mg^2_{t_1,c} + mg^2_{t_2,c} + \ldots} \text{ is the norm of the class.}$$

Since the concept of class is generally imprecise, the class assignment of an object cannot be exactly “true” or “false”. It is also impractical to categorize an object to one class only while an object may is conceptually relative to several classes. Therefore, for an input object, ACIRD Classifier gives the best N classes that are closest to the intension 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.

Experiment Results

At the time we perform this experiment, there are totally 512 classes in L_{ACIRD} with 386 most specific classes. (i.e. the leave nodes of the class hierarchy). Yam provides 9,778 training objects and 8,855 testing objects, which have been manually classified to the classes in L_{ACIRD}. The training set and the testing set are disjoint. Before the learning process, ten human experts extract the keywords from each class as the classification knowledge benchmark, denoted as $Know_c^U$. The learning and testing processes run on $Know_c^U$, $Know_c^*$, and $Know_c$, which are marked as "10 Users", "With PTS", and "Without PTS", respectively, in Fig. 8. The result shows that $Know_c^*$ has the quality in par with the manually extracted classification knowledge $Know_c^U$ in terms of the accuracy of class assignment of objects.

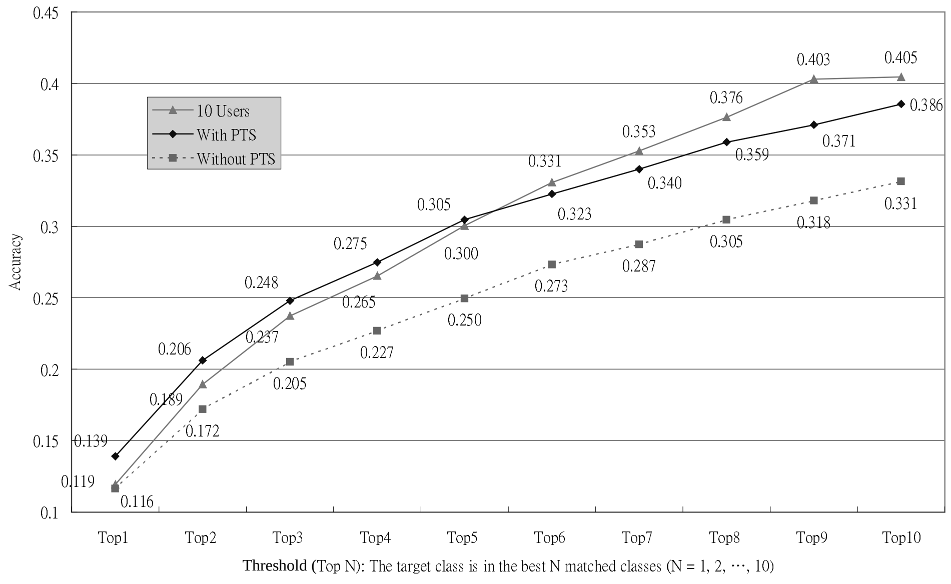


Fig. 8. The classification accuracy of 8855 testing objects based on exact class match.

However, the classification accuracy of all the three cases is not high enough. By analyzing the training and testing sets, we found that there are not sufficient training objects in many classes, and some training/testing objects contains very few keywords because they are non-text pages or link-only pages. Thus, we design another experiment to circumvent the situation. The same testing process is performed based on the 12 most general classes[6] of Yam, and the resulting classification accuracy is shown in Fig. 9. The "**Top 1**" accuracy of $Know_c^*$ is increased from 0.139 in Fig. 8 to

[6] In http://taiwan.iis.sinica.edu.tw/en/yam/, users can see the 12 most general categorizes of Yam: "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".

0.595. The increase is due to sufficient training objects in the most general classes, and the total number of testing classes is reduced from 512 to 12 so that the noise level is also reduced.

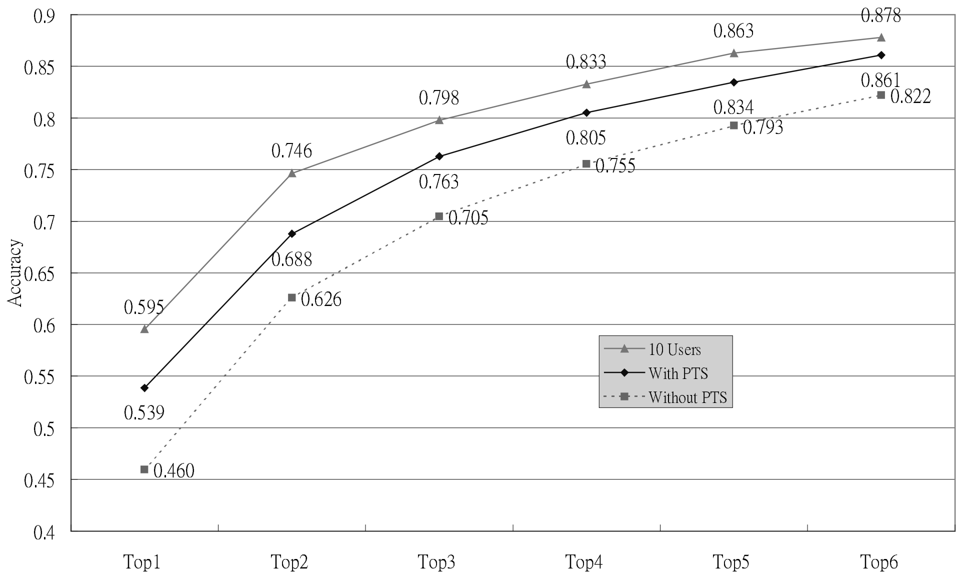


Fig. 9. The classification accuracy of 8855 testing objects based on 12 most general classes.

The numbers of objects and keywords of the twelve most general classes are shown in Table 2. As we can observe, the distribution of the numbers is skewed, and some classes still suffer from the problem of insufficient training objects and keywords. Thus, we perform another experiments on classes with sufficient training objects and keywords only. In the experiment, every tested class has at least 40 training objects. The total tested classes are reduced from 512 to 48 in contrast with the experiment shown in Fig. 8. Since there are sufficient training objects in the classes, we call these classes as well-trained classes and their refined classification knowledge as well-trained classification knowledge. To evaluate the quality of well-trained classification knowledge, the classification knowledge generated from the ten human experts is compared. The results in Fig. 10 shows that our learning model is capable of discovering more accurate classification knowledge than that of human experts, if there are sufficient training objects in the classes. As shown in the figure, the "**Top N**" classification accuracy is also dramatically increased while there are sufficient training objects.

Table 2. The distribution of training objects in Yam's 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%)

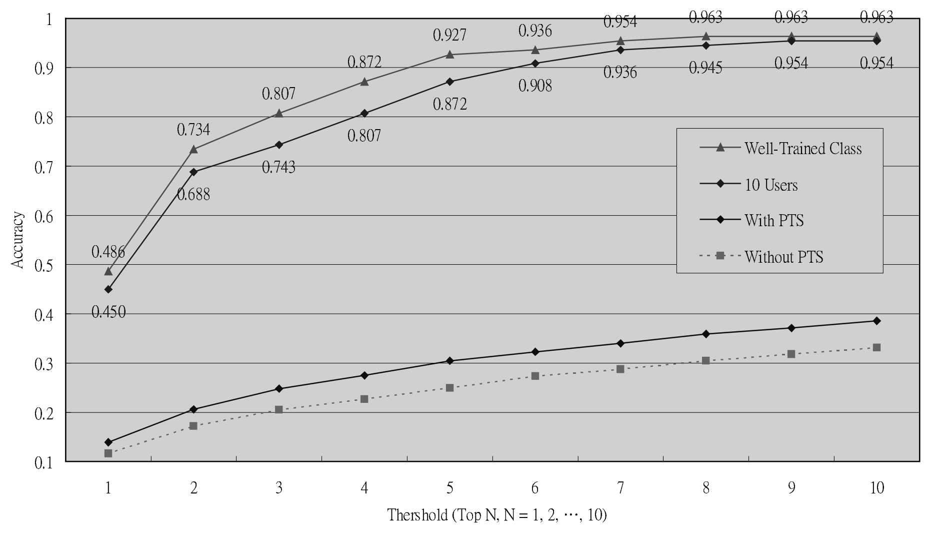


Fig. 10. The classification accuracy of testing objects on the well-trained classification knowledge in comparison with the knowledge generated by human experts.

7. Two-Phase Search Engine

Most current search engines return user queries with a list of ranked documents that is time consuming and inconvenient for users to access the needed documents. In ACIRD, the system provides *two-phase search* that allows users to perform class-level search and object-level search. Utilizing the two-phase search, the user can associate his information needs to the classes in the hierarchy, navigate the class hierarchy, find the interesting classes, and finally retrieve the documents from the designated class. The above procedure can be carried out repeatedly until the desired information is accessed. To determine the effectiveness of the two-phase search, it is critical that the terms of user queries are in $Know_c^*$ so that the class-level search on $Know_c^*$ can return the desired classes. The conjecture "most query terms are in $Know_c^*$" is further investigated by the following analysis on the query log of Yam.

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 terms from queries and count their frequency. By regarding each term as an information need and its frequency as the reference count, 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. A series of tests on the distribution of keywords of $Know_c^*$ are performed to verify the above conjecture. Each test has a different threshold to select keywords for $Know_c^*$, denoted by "Th = x.x". If the keywords in $Know_c^*$ are the same with the terms in CT_{Yam}, their reference counts are the reference counts of the terms; otherwise, the reference count of the keyword is 0. The summation of the reference counts and the number of keywords of each test are shown in Table 3. Regarding references and query terms of Yam's query log as the baseline, the recall rate and index rate of each test are defined as:

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

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

For example, in the case "Th = 0" (i.e., no keyword is eliminated), indexed keywords cover 96.92% information needs (the recall rate) with about doubled index size of query terms (the index rate). However, when "Th = 0.5", it covers 69.89% information needs with about 30.91% index size. In practical, it is an acceptable compromise. From the experiments, it can indicate that two-phase search is capable to shrink the searching domain with little information loss and can be applied as an efficient search engine.

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

Based Line: Query Terms of Yam	648006 (100%)	9644 (100%)
Threshold to Filter Out Keywords in Refined Classification Knowledge	References of Information Needs (Recall Rate)	Number of Keywords (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%)

Two-Phase Search Algorithm

Based on the above analysis, 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 most queries efficiently and returns with structured presentation. The conventional searching approach, i.e. search all objects in the object-level search, is also implemented as an "escape" for users. After quickly reviewing the result of class-level search, the user can navigate down the class hierarchy or choose object-level search in some class. The algorithm of Two-Phase Search is described in the following, and the data flow diagram is shown in Fig. 11.

Two-Phase Search Algorithm

1. **Process query string**: Parsing the query string into a sequence of keywords indicated by keyword ID (KID).
2. **Class-Level Search**: Retrieving classes (CID) associated to the KID, calculating each class's relevance score, and sorting CID's by the scores in descending order. Presenting the result in HTML format.
3. **Object-Level Search in a class**: Retrieving objects, in the selected class, associated to the KID, calculating each object's relevance score, and sorting OID's by the scores in descending order. Presenting the result in HTML format.
4. If Two-Phase Search was unable to find the desired information, then the user can chose the conventional search approach to search all objects.
5. **Search all objects**: Retrieving all objects associated to the KID, calculating each object's relevance score, and sorting OID by MG in decreasing order. Presenting the result in HTML format.

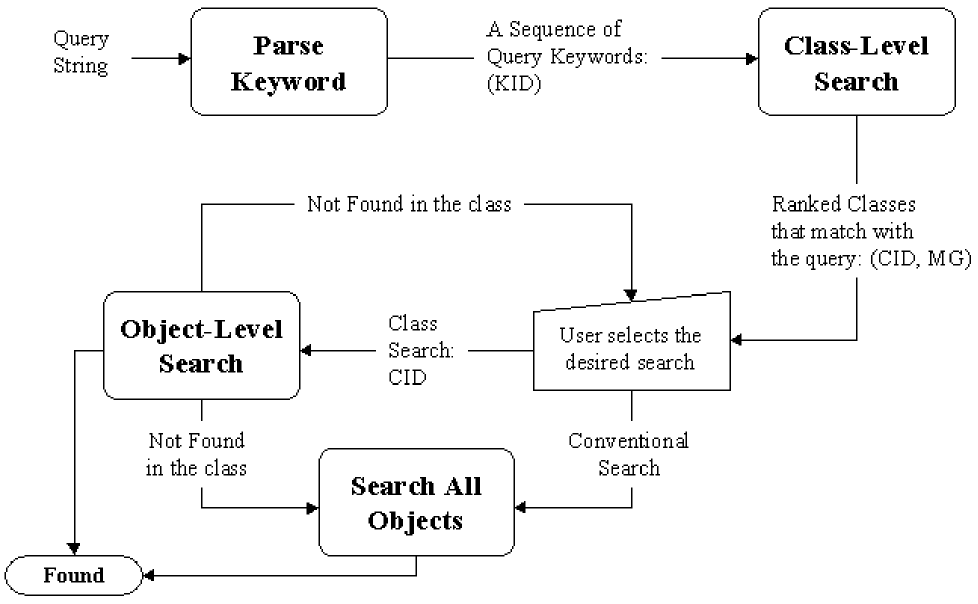
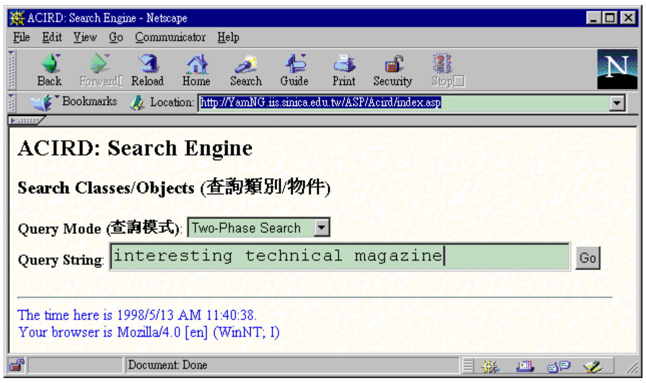


Fig. 11. Data flow diagram of Two-Phase Search algorithm.

Examples of Two-Phase Search Engine

Users can search desired objects from ACIRD[7] by giving their query strings. For example, in Fig. 12, the user selects the query mode “Two-Phase Search” and gives the query “interesting technical magazine”.



ACIRD: Search Engine

Search Classes/Objects (查詢類別/物件)

Query Mode (查詢模式): Two-Phase Search

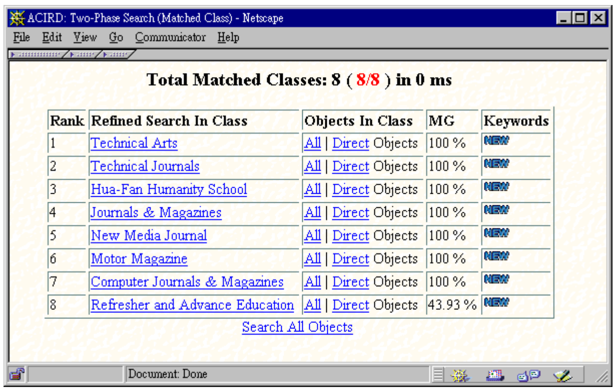
Query String: interesting technical magazine

The time here is 1998/5/13 AM 11:40:38.
Your browser is Mozilla/4.0 [en] (WinNT; I)

ACIRD: Search Engine - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://YamNG.iis.sinica.edu.tw/ASP/Acird/index.asp
Go
Document: Done

Fig. 12. Two-Phase Search in ACIRD (Query Interface).

The result of the query is shown in Fig. 13, which contains the following information. “*Rank*” indicates the ranking order of the matched classes. “*Refined Search in Class*” presents the class name that user can resume the same query on the class objects. “*Object In Class*” shows two links, “All” and “Direct”. The former, shown in Fig. 14, lists all objects in the class (including objects of its subclasses); the latter, shown in Fig. 15, lists the class’s direct objects. “*MG* (*membership grade*)” indicates the normalized relevance score. “*Keywords*” links to the keywords of $Know_c^*$. The page in Fig. 13 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 interesting to the user.



Total Matched Classes: 8 (8/8) in 0 ms

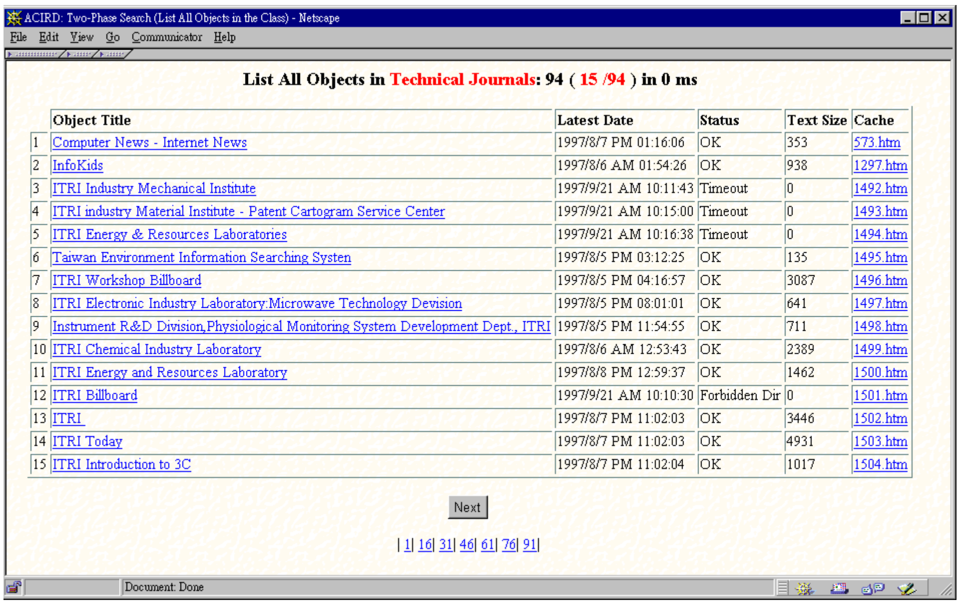
Rank	Refined Search In Class	Objects In Class	MG	Keywords
1	Technical Arts	All | Direct Objects	100 %	NEW
2	Technical Journals	All | Direct Objects	100 %	NEW
3	Hua-Fan Humanity School	All | Direct Objects	100 %	NEW
4	Journals & Magazines	All | Direct Objects	100 %	NEW
5	New Media Journal	All | Direct Objects	100 %	NEW
6	Motor Magazine	All | Direct Objects	100 %	NEW
7	Computer Journals & Magazines	All | Direct Objects	100 %	NEW
8	Refresher and Advance Education	All | Direct Objects	43.93 %	NEW

Search All Objects

ACIRD: Two-Phase Search (Matched Class) - Netscape
File Edit View Go Communicator Help
Document: Done

Fig. 13. Two-Phase Search in ACIRD (Query Result: Matched Classes).

[7] The current version of ACIRD (http://YamNG.iis.sinica.edu.tw/Acird/class.htm) provides Chinese interface only. The figures shown in this example are the English translations.

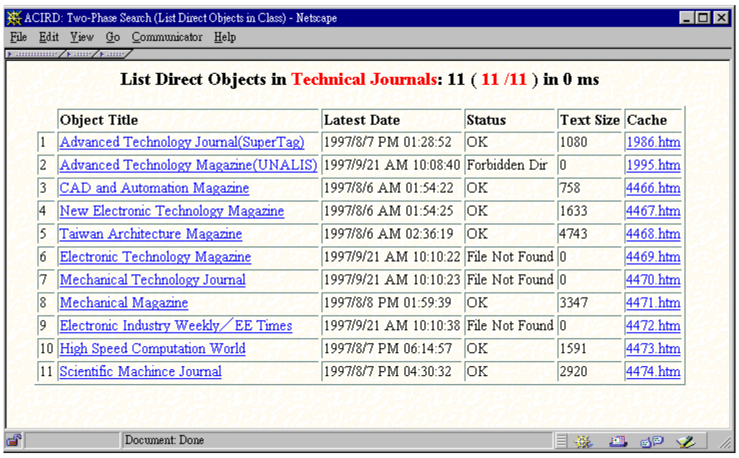

ACIRD: Two-Phase Search (List All Objects in the Class) - Netscape
File Edit View Go Communicator Help
Next
Document: Done

List All Objects in Technical Journals: 94 (15 /94) in 0 ms

	Object Title	Latest Date	Status	Text Size	Cache
1	Computer News - Internet News	1997/8/7 PM 01:16:06	OK	353	573.htm
2	InfoKids	1997/8/6 AM 01:54:26	OK	938	1297.htm
3	ITRI Industry Mechanical Institute	1997/9/21 AM 10:11:43	Timeout	0	1492.htm
4	ITRI industry Material Institute - Patent Cartogram Service Center	1997/9/21 AM 10:15:00	Timeout	0	1493.htm
5	ITRI Energy & Resources Laboratories	1997/9/21 AM 10:16:38	Timeout	0	1494.htm
6	Taiwan Environment Information Searching Systen	1997/8/5 PM 03:12:25	OK	135	1495.htm
7	ITRI Workshop Billboard	1997/8/5 PM 04:16:57	OK	3087	1496.htm
8	ITRI Electronic Industry Laboratory:Microwave Technology Devision	1997/8/5 PM 08:01:01	OK	641	1497.htm
9	Instrument R&D Division,Physiological Monitoring System Development Dept., ITRI	1997/8/5 PM 11:54:55	OK	711	1498.htm
10	ITRI Chemical Industry Laboratory	1997/8/6 AM 12:53:43	OK	2389	1499.htm
11	ITRI Energy and Resources Laboratory	1997/8/8 PM 12:59:37	OK	1462	1500.htm
12	ITRI Billboard	1997/9/21 AM 10:10:30	Forbidden Dir	0	1501.htm
13	ITRI	1997/8/7 PM 11:02:03	OK	3446	1502.htm
14	ITRI Today	1997/8/7 PM 11:02:03	OK	4931	1503.htm
15	ITRI Introduction to 3C	1997/8/7 PM 11:02:04	OK	1017	1504.htm

| 1| 16| 31| 46| 61| 76| 91|

Fig. 14. Two-Phase Search in ACIRD (Query Result: List all objects under a class).

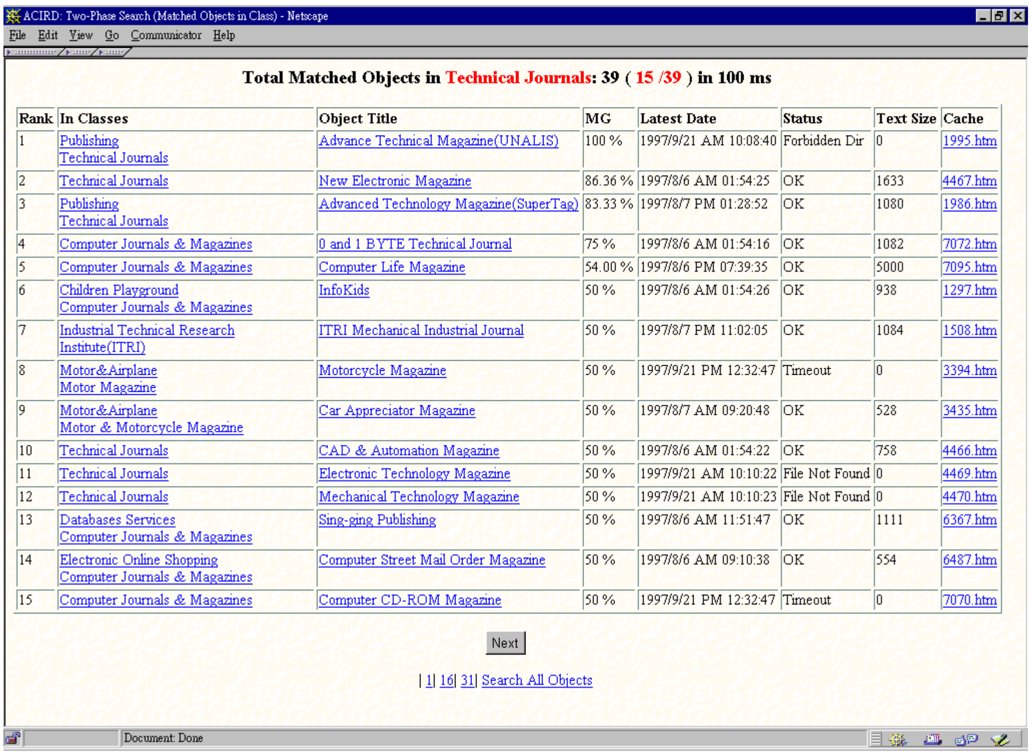

ACIRD: Two-Phase Search (List Direct Objects in Class) - Netscape
File Edit View Go Communicator Help
Document: Done

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.15. Two-Phase Search in ACIRD (Query Result: List direct objects of a class).

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. 16. "*In Class*" presents one or several classes that contain the object. It can be clicked to list all objects in the class. "*Object Title*" shows the object's content in <TITLE>, which 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. "Text Size" is the file length of the object's HTML source without counting non-text media. The user can quickly preview the page by clicking on the link in the column "*Cache*".


ACIRD: Two-Phase Search (Matched Objects in Class) - Netscape
File Edit View Go Communicator Help
Next
Document: Done

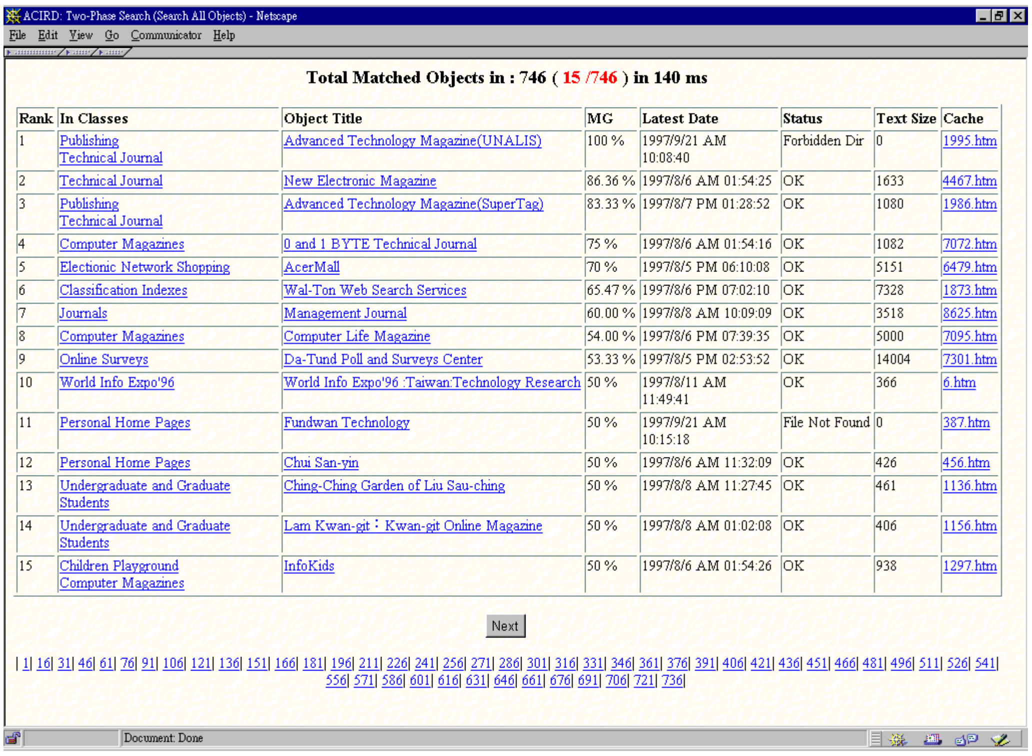
Total Matched Objects in Technical Journals: 39 (15 /39) in 100 ms

Rank	In Classes	Object Title	MG	Latest Date	Status	Text Size	Cache
1	Publishing 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	86.36 %	1997/8/6 AM 01:54:25	OK	1633	4467.htm
3	Publishing 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	7072.htm
5	Computer Journals & Magazines	Computer Life Magazine	54.00 %	1997/8/6 PM 07:39:35	OK	5000	7095.htm
6	Children Playground Computer Journals & Magazines	InfoKids	50 %	1997/8/6 AM 01:54:26	OK	938	1297.htm
7	Industrial Technical Research Institute(ITRI)	ITRI Mechanical Industrial Journal	50 %	1997/8/7 PM 11:02:05	OK	1084	1508.htm
8	Motor&Airplane Motor Magazine	Motorcycle Magazine	50 %	1997/9/21 PM 12:32:47	Timeout	0	3394.htm
9	Motor&Airplane Motor & Motorcycle Magazine	Car Appreciator Magazine	50 %	1997/8/7 AM 09:20:48	OK	528	3435.htm
10	Technical Journals	CAD & Automation Magazine	50 %	1997/8/6 AM 01:54:22	OK	758	4466.htm
11	Technical Journals	Electronic Technology Magazine	50 %	1997/9/21 AM 10:10:22	File Not Found	0	4469.htm
12	Technical Journals	Mechanical Technology Magazine	50 %	1997/9/21 AM 10:10:23	File Not Found	0	4470.htm
13	Databases Services Computer Journals & Magazines	Sing-ging Publishing	50 %	1997/8/6 AM 11:51:47	OK	1111	6367.htm
14	Electronic Online Shopping Computer Journals & Magazines	Computer Street Mail Order Magazine	50 %	1997/8/6 AM 09:10:38	OK	554	6487.htm
15	Computer Journals & Magazines	Computer CD-ROM Magazine	50 %	1997/9/21 PM 12:32:47	Timeout	0	7070.htm

| 1| 16| 31| Search All Objects

Fig.16. Two-Phase Search in ACIRD (Query Result: Search Objects in Specific Classes).

If the user presses the link to perform “Search All Objects”, the result is shown in Fig. 17. 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 a large number of links.


ACIRD: Two-Phase Search (Search All Objects) - Netscape
File Edit View Go Communicator Help
Next
Document: Done

Total Matched Objects in : 746 (15 /746) in 140 ms

Rank	In Classes	Object Title	MG	Latest Date	Status	Text Size	Cache
1	Publishing Technical Journal	Advanced Technology Magazine(UNALIS)	100 %	1997/9/21 AM 10:08:40	Forbidden Dir	0	1995.htm
2	Technical Journal	New Electronic Magazine	86.36 %	1997/8/6 AM 01:54:25	OK	1633	4467.htm
3	Publishing Technical Journal	Advanced Technology Magazine(SuperTag)	83.33 %	1997/8/7 PM 01:28:52	OK	1080	1986.htm
4	Computer Magazines	0 and 1 BYTE Technical Journal	75 %	1997/8/6 AM 01:54:16	OK	1082	7072.htm
5	Electionic Network Shopping	AcerMall	70 %	1997/8/5 PM 06:10:08	OK	5151	6479.htm
6	Classification Indexes	Wal-Ton Web Search Services	65.47 %	1997/8/6 PM 07:02:10	OK	7328	1873.htm
7	Journals	Management Journal	60.00 %	1997/8/8 AM 10:09:09	OK	3518	8625.htm
8	Computer Magazines	Computer Life Magazine	54.00 %	1997/8/6 PM 07:39:35	OK	5000	7095.htm
9	Online Surveys	Da-Tund Poll and Surveys Center	53.33 %	1997/8/5 PM 02:53:52	OK	14004	7301.htm
10	World Info Expo'96	World Info Expo'96 :Taiwan:Technology Research	50 %	1997/8/11 AM 11:49:41	OK	366	6.htm
11	Personal Home Pages	Fundwan Technology	50 %	1997/9/21 AM 10:15:18	File Not Found	0	387.htm
12	Personal Home Pages	Chui San-yin	50 %	1997/8/6 AM 11:32:09	OK	426	456.htm
13	Undergraduate and Graduate Students	Ching-Ching Garden of Liu Sau-ching	50 %	1997/8/8 AM 11:27:45	OK	461	1136.htm
14	Undergraduate and Graduate Students	Lam Kwan-git : Kwan-git Online Magazine	50 %	1997/8/8 AM 01:02:08	OK	406	1156.htm
15	Children Playground Computer Magazines	InfoKids	50 %	1997/8/6 AM 01:54:26	OK	938	1297.htm

| 1| 16| 31| 46| 61| 76| 91| 106| 121| 136| 151| 166| 181| 196| 211| 226| 241| 256| 271| 286| 301| 316| 331| 346| 361| 376| 391| 406| 421| 436| 451| 466| 481| 496| 511| 526| 541| 556| 571| 586| 601| 616| 631| 646| 661| 676| 691| 706| 721| 736|

Fig. 17. Two-Phase Search in ACIRD (Query Result: Search All Objects).

8. Conclusions and Future Work

In this paper, we shows that machine learning and data mining techniques can be applied to learn and refine classification knowledge. Based on the knowledge, automatic classification is able to automatically classify the Internet documents to classes in a class hierarchy. According to the analysis of the query log of Yam, we show that the knowledge can be the *meta-index* to shrink the searching domain. The index is used to retrieve classes containing potentially desired documents efficiently. The result of class-level search can be presented in a comprehensible view of concepts. Users can associate the queries to the presented classes and perform object-level search in the classes to get their desired documents. In this way, the system helps users from visiting and finding information from a large number of ranked documents.

In the future, the learning model must be extended to the incremental learning model to cope with dynamic changes of the Internet. In addition, the classification accuracy of the learning methods and the classifier still has room for improvement. There are also some issues for further study. For example, extending the research of mining term associations in classes to automatic construction of the thesaurus, which is corresponding to the semantics of terms in the specific domain. Also, by analyzing the query log of Yam, the system can learn and extract new terms that can not be found in thesauruses, such as "MP3", "ICQ", "CGI", etc. to extend the term-base of ACIRD.

9. References

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

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

[3] K. Spark 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.

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

[5] 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.

[6] B. Ford and R. M. J. Iles, "The What and Why of Problem Solving Environments for Scientific Computing," Problem Solving Environment for Scientific Computing, Ford and Chatelin, Eds, Elsevier Science Publishing Co., 1987, pp. 3-22.

[7] 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.

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

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

[10] 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.

[11] 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.

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

[13] 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.

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

[15] M. C. Chen and J. M. Ho, "An Automatic Classifier and Explore for Internet Resource", NSC Technical Report.

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

[17] 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.

[18] 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.

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

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

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

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

[23] 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.

[24] 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.

[25] G. Connet, "Unstructured Data Bases or Very Efficient Text Searching", ACM PODS, Vol. 2, pp. 117-124.

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

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

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

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

[30] 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.

[31] 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.

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

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

[34] 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.

[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] B*tree

10.