ACIRD: An Intelligent Internet Information System Based on Data Mining (Extended Abstract)

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

1. Introduction

The explosive growth of the Internet dramatically changes the way of working and living that the Internet becomes a major source of information. However, the excessive information on the Internet creates the information overflow problem. As a result, information retrieval (IR) systems (or search engines) come to help the Internet users to alleviate the problem. The conventional IR systems are designed to facilitate rapid retrieval of information for diverse users. By applying keyword-based index/search approaches [12], keywords in a document are extracted, stored and indexed in databases. Then the indexes are employed in retrieving documents relevant to a query represented by keywords.

From the perspective of retrieval efficiency, keyword-based IR systems are useful to handle a large document-base. However, documents collected from the Internet are extremely numerous. In such case, for a query with two words[1] submitted to a search engine implemented with vector space model (VSM) [8, 9], thousands of documents are probably retrieved. For example, the query "education and university", there are 7379086 hits by Infoseek, 2879 hits by Yahoo, and 237902 hits by WebCrawler. Ranking a large number of documents using one or two keywords can not order the documents effectively as the preference of the user. Consequently, user has to read many undesired documents before obtaining the needed information.

The conception gap between Web pages developers and Internet users enlarges the difference between the retrieve results from user's expectation. Due to the richness of language and culture, developers and users may use different terms to represent the same semantics, or use same terms to describe different meanings. Therefore, in conventional IR systems, desired documents are probably not selected. For instance, the term "airline schedule" in documents does not fully match the term "airplane schedule" in query, but both terms represent the same semantics. This is another reason why many IR systems may retrieve thousands of documents only with few desired. For a specific IR environment, a thesaurus database with terms for the special domain knowledge can alleviate such problem. However, because of the diverse information in the Internet, no static thesauruses can cover the mismatching and also shifting semantics of terms. Therefore, the Internet IR systems should be able to search relevant documents efficiently, rank and organize the documents in accordance with user's expectation.

Our system, *ACIRD*[2] *(*Automatic Classifier for the Internet Resource Discovery) [11] is designed to automatically classify Chinese and English documents to proper classes in a class hierarchy used in Yam[3], and to reply user Internet query with a friendly and expressive answer. *ACIRD* learns classification knowledge from the documents collected and classified in Yam. *ACIRD* also mines the association rules among terms and infers the term associations to refine the classification knowledge in each class. Based on the discovered classification knowledge, *ACIRD* implements a two-phase search to shrink the search domain in answering user query, and presents a hierarchically navigable result to user.

In the rest of the extended abstract, the *ACIRD* system is described in Section 2. In Section 3, the learning model is presented to show the details of ACIRD. Experiments of the classification process are shown in Section 4. Finally, we conclude the extended abstract.

2. The *ACIRD* System

Before describing the overview of the system, we explain the following terminology used throughout the paper.

- *Class* corresponds to the Yam's category. The classification knowledge of a class ($Know_c$) is represented by a set of terms that each one has its support value.
- *Object* corresponds to the Internet HTML document. The object knowledge ($Know_o$) is also a set of terms.
- *Term* is the word or phrase extracted from an object or generalized to a class by the learning process.
- *Support* is the improtance degree of a term to some object or class. The value is normalized to [0, 1].
- *Keyword* corresponds a representative term.
- *Memership grade* (*MG*) is the supporting degree of a keyword to some object or class as *support* is the supporting degree of a term.

ACIRD is motivated to improve the poor performance of the current manual HTML document classification process. Yam's categories is used as the class hierarchy of *ACIRD* called *ACIRD Lattice* (L_{ACIRD}) [14]. The learning includes two processes, the training process and testing process. The training process learns $Know_c$ of each

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

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

[3] http://www.yam.org.tw/b5/yam, a very popular local search engine in Taiwan.

class in L_{ACIRD}. Then the testing process verifies the correctness of $Know_c$ by comparing the assigned class of newly collected documents (testing data) with the class manually categorized by human experts. Based on $Know_c$, *ACIRD Classifier* automatically classifies new incoming Internet documents based on the similarity match of VSM. In addition, *ACIRD* has *Two Phase Search Engine* to improve search performance and organize the query results.

In the rest of the section, we give an overview of the system based on issues of IR systems in [12]: *document operation*, *conceptual model*, *term operation*, *file structure*, *query operation*, and *hardware*.

Document Operation

Each document (object) is assigned a unique OID in the database. An object can be instantiated from several classes according to Yam's current categorization (i.e., L_{ACIRD}). Before performing the classification process, objects are preprocessed into terms with supports to form the object's feature vector (i.e. $Know_o$*)*.

Conceptual Model

A probabilistic document conceptual model is used. $Know_o$ is represented by a collection of terms with supports normalized to [0, 1]. Learning from the training objects of a class, $Know_c$ of the class is also represented by a feature vector with terms and supports. Based on a predefined threshold θ_C of class, terms in $Know_c$ are divide into two types.

- *Representative*: term's support is not less than θ_C. Representative term is regarded as *keyword*.
- *Non-representative*: term's support is less than θ_C.

For each class, association rules between terms are mined. The rule is named *term association*. Based on the digraph constructed from term associations and term supports to a class, an inference model of terms to the class is proposed to refine the support of a term to *promote a non-representative term to representative one* [7]. Term associations are also employed to refine $Know_c$ as *refined classification knowledge* ($Know_c^*$).

Term Operation

Terms in an object are extracted according to our Chinese Term-Segmentation Rules [15] associated with a term-base, which is the result from analyzing the past query logs of Yam. Term operation is involved in Document Operation and Query Operation.

File Structure

In the system, inverted indexes of terms in classes and objects are implemented by using the relational database system. Given a term, relevant classes (indicated by CID) and objects (indicated by OID) can be retrieved efficiently.

Query Operation

A query is composed of a set of terms that similarity match based on VSM is applied to retrieve relevant objects or classes. Using $Know_c$ of classes in L_{ACIRD}, the Two-Phase Search Engine [14] performs class-level search in the first phase. Matched classes form a shrunk view of L_{ACIRD} to reduce the searching domain of the following object-level search. In the second phase, for each matched class, terms in the query are employed to match objects in the class. By integrating qualified classes and matched objects in these class, a tree (or lattice) with probabilities in branches is shown to the user. Thus, the two-phase search mechanism prunes the searching space and presents a structured view to users.

Hardware

The currently system is implemented on a Pentium II 266 and 256M SDRAM machine with NT Server 4.0 (SP3) and SQL Server 6.5 (SP1) software systems.

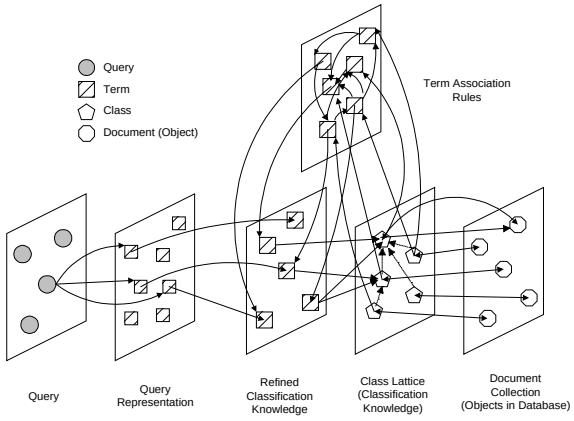

Fig. 1. Two-phase Query Process in ACIRD

In Fig. 1, we summarize the overview of the system. First, *Documents* and *Class Lattice* (i.e., L_{ACIRD}) are obtained from Yam. Then, terms and their supports are extracted from documents to generalize *Classification Knowledge* (i.e., $Know_c$). Finally, $Know_c$ is refined to *Refined Classification Knowledge* (i.e., $Know_c^*$) by using the mined *Term Association Rules* in the class. The Two-Phase Search Engine parses *Query* into term-based *Query Representation* that is used to match classes and documents.

3. The Learning Model

In this section, we describe the design and implementation of ACIRD in details. ACIRD applies supervised learning techniques by regarding documents that are manually assigned with one or more classes as the training set. In the experiments, there are 9778 objects in 512 classes. The learning model is divided into four processes.

3.1 Preprocessing and Knowledge Representation

To determine the value of the features that are used to represent the individual object within a class, two modules are implemented.

HTML Parser

HTML Parser is implemented to extract *sentences* included within HTML tags and to determine the *weighted value* of each sentence. An HTML document has many HTML tags, such as TITLE, Hn (headings), B, I, U, etc. Additionally, META tag gives information about documents such as "classification" and "keyword". Apparently, HTML tags provide significant information to index and classify the documents. Currently, the weights of HTML tags are defined by experienced Webmasters to indicate the importance of tag. The weights are stored as a table in the database for easy accesses and updates. The system categorizes HTML tags into four types.

- *Informative*. Sentences included in tags, such as Hn, B, I, and U, have high infromation values.
- *Uninformative*: Sentences enclosed by tags, such as AREA, COL, COMMENT, etc., are not processed and indexed.
- *Statistical*: Sentences included in tags, like !DOCTPYE, APPLET, OBJECT, SCRIPT, etc., are extracted and stored in database for the statistical purpose in the future.
- *Skippable*: Tags, such as BR and P, have no effects and are omitted.

Term Processor

Term Processor is applied to extract terms in the sentence and to count *term frequency*. This is essentially the dictionary creation process. Based on the pre-constructed term-base and term-segmentation rules, Term Processor deals with the ambiguous segmentation of terms and extracts terms from a sentence [15]. After a term is extracted from an object, the support sup of the term to the object is defined in equation (3.1) to indicate the importance of the term. The value is normalized to [0, 1].

$$sup'_{t_i,o} = \sum_{T_j} tf_{ij} \cdot w_{T_j},\ t_i \text{ is a term in the sentence enclosed by TagPattern } T_j,$$
$$tf_{ij} \text{ is the term frequency of } t_i \text{ in } T_j, \text{ and} \quad (3,1)$$
$$w_{T_j} \text{ is the maximal weighed tag in } T_j.$$

$$sup_{t_i,o} = \frac{sup'_{t_i,o}}{\underset{t_i \text{ in } o}{MAX}(sup'_{t_i,o})}, \text{ i.e., } sup \text{ is normalized into } [0,1]$$

As shown in equation (3.1), a term extracted from an object has sup . A training object can be represented as a vector of terms with their normalized values between 0 and 1. The vector representation is also used in the similarity match based on the VSM.

3.2 Feature Selection

The computional complexity of inducing process is generally exponentially increased by the dimension of features. In addition, excessive features increase the level of noise. Thus, feature selection is essential to alleviate the problem. In the training set, the average number of terms is 28.64, which is too large to generalize terms from objects to the class. To reduce the size of dimension of training examples, a threshold test is applied for feature selection. For each object, extracted terms are examined against pre-defined threshold of support θ_s (for example, θ_s = 0.2). Terms not filtered out are collected into $Know_o$. In this way, the problem of feature selection is shifted to the selection of θ_s . Higher θ_s , more terms will be filtered out, and reamining few terms can not represent the object's knowledge. In contrast, with lower θ_s , feature selection can not be effective. After processing all training objects and analyzing the distribution of ranges of term supports shown in Fig. 2, we observed that more than one half of terms locate in the range [0, 0.2). Therefore, in the feature selection of objects, the system uses $\theta_s = 0.2$ to filter out terms with low supports. In average, the number of terms in an object is reduced from 28.64 to 11.61.

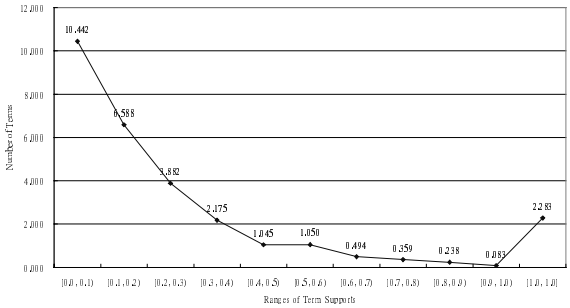

Fig. 2. The range distribution of term supports to all training objects

3.3 Induction Process

To find $Know_c$, the induction process generalizes terms from objects to their associated classes. After parsing and selecting terms from training objects, the induction process is applied to each class from specific level (BOTTOM) to general level (TOP). The *class-assignment* of a training object is based on the assumption that Yam's manual categorization is correct. In conventional Boolean IR systems, the value of support of a term to an object is either TRUE or FALSE, and the generalization of term t to class c is based on the occurrence of t in the objects of the class. That is, the generalization is according to the term's document frequency in the class. However, in ACIRD, the term supports to an objects is ranged from 0 to 1 that meet the intuition that terms in an object are not equally significant. Term's document frequency in the class and term's support should be considered simultaneously. Thus, the system calculates the summation of a term's supports to objects in the class, and the value, denoted by $sup_{t,c}$, is called *term support to the class*. In the same way with

equation (3.1), $sup_{t,c}$ is also norlmaized to [0, 1] as shown in equation (3.2).

$sup'_{t_i,c} = \sum_{o_j} sup_{t_i,o_j}$, sup_{t_i,o_j} is the term support of t_i to o_j, o_j is an object in the class c. (3.2)

$sup_{t_i,c} = \frac{sup'_{t_i,c}}{MAX(sup'_{t_i,c})}$, i.e., $sup'_{t_i,c}$ is normalized into $sup_{t_i,c}$ ranged [0,1] (4.2)

Likewise, terms with supports in a class form $Know_c$ as terms with supports to an object represent $Know_o$. $Know_c$ can be represented as the feature vector estimated from objects' vectors.

By analyzing the term distributions in each class, we discovered that most terms' supports locate in low support range (e.g., [0, 0.3)). The result is shown in Fig. 3. Thus, feature selection process is also necessary to refine $Know_c$.

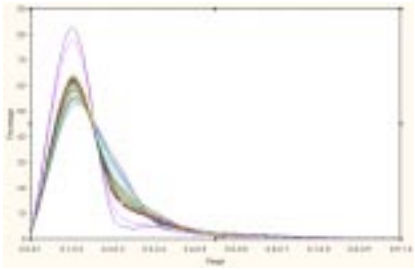

Fig. 3. Term supports distributions of classes.

3.4 Knowledge Refining Process

As shown in Fig. 3, if we apply the same feature selection process in section 3.2, almost all terms in $Know_c$ will be filtered out because of very low supports. Hence, before feature selection process, we apply *mining term associations* and *perfect term support algorithm* to refine the extracted knowledge.

Mining Term Associations

Two important issues should be considered before applying mining association rules [16,17,18]. One is what granularity should be used to mine associations. The other is what boundary should be decided to generate association rules that is the *transaction database* defined in [17].

- *The granularity of mining associations*. In [19], authors restricted the granularity of generating associations to 3-10 sentences per paragraph. It is infeasible since a paragraph may have hundreds of sentences. Moreover, the importance of a sentence of Internet document depends on the associated HTML tags. Therefore, we can regard *informative sentences* in a document as transactions defined in [17].
- *The boundary of generating association rules*. As Internet documents are published by a diverse people, the semantics of words is different from case to case. In the system, we apply mining term associations to the documents in a class separately rather than all the documents in the database.

Based on the arrangements, we translate our problem domain into the domain of mining association rules [17] by regarding (i) *terms* are corresponding to *items*; (ii) *documents* in the class are corresponding to *transactions*; (iii) The *class* is corresponding to the *transaction database*.

Concentrating on documents of a class instead of all classes takes the advantage of small database size, especially that the complexity of mining associations is generally exponentially increased with the size of the database. If the size of database is not very large, a simple mining algorithm, such as Apriori [16], can be efficiently applied to our system.

Perfect Term Support Algorithm (PTS[4])

The *confidence* defined in [17] is used to promote *non-representative* terms to *keywords* by refining $Know_c$ to $Know_c^*$. For example, applying mining term associations in the class "Art", the rule "exhibition → art" is obtained with confidence and support are 0.826 and 0.1. Assume an association rule with 10% supports is considered useful. After applying the rule to refine $Know_c$, which includes $sup_{exhibition,Art}$ is 0.13, and $sup_{art,Art}$ is 1, $sup_{exhibition,Art}$ is promoted from 0.13 to 0.826[5].

Based on the original support, "exhibition" will be filtered out due to its low support (0.13). However, the promoted value (0.826) is enough to support "exhibition" as a keyword. In [7], the simulation of PTS's effectiveness is shown in Table 1. Top 10 and 20 mean only the top 10 and 20 features are selected.

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

	Without PTS algorithm		With 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

4. Evaluation of Automatic Classification

The automatic classification of new objects is based on VSM. By regarding keywords with *MG*s in $Know_c^*$ and terms with supports in $Know_o$ as feature vectors, the classifier estimates the relevance scores between a new object and a class. Because of the classes are not mutually exclusive, the assignment of object to classes is not unique. Actually, there are totally 512 classes in L_{ACIRD} with 386 basic classes[6]. It is inappropriate to categorize an object to the exact class while object covers concepts belonging to several classes. Thus, the

[4] Due to the restriction of paper length, we omit the details of PTS, which can be found in [7].

[5] $sup^*_{exhibition,Art} = confidence_{exhibition \to art} \times sup_{art,Art}$ = 0.826*1 = 0.826 $\geq sup_{exhibition,Art}$ = 0.13.

[6] According to the definition in [14], a basic class is the most specific class in the lattice.

classification accuracy is estimated by the criterion that if the target class is located in the set of best N matched classes. The criterion is called "TopN".

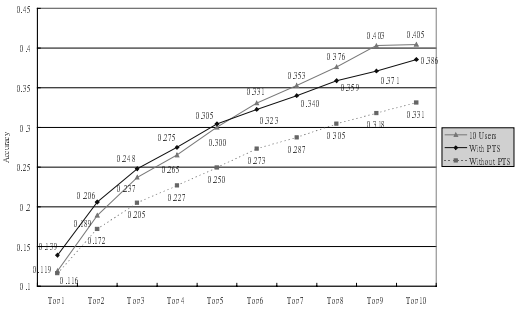

Fig. 4. The classification accuracy of 8855 testing objects.

During the design and implementation of ACIRD, it uses 8855 objects manually classified to the 512 classes in L_{ACIRD} from Yam as testing samples. To evaluate the correctness of $Know_c^*$, keywords of each class are manually extracted by 10 users ($Know_c^U$) to form the benchmark of each class's classification knowledge. The same testing process runs based on $Know_c^U$, $Know_c^*$, and $Know_c$ are indicated as "10 Users", "With PTS", and "Without PTS" shown in Fig. 4. The result shows that our learning methods can learn classification knowledge $Know_c^*$ that is close to human classification knowledge $Know_c^U$.

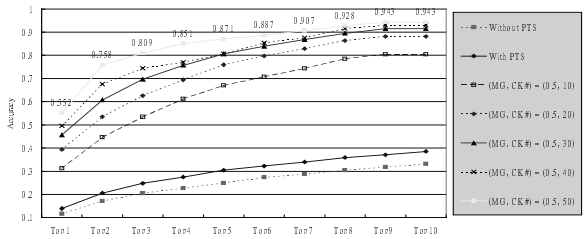

Fig. 5. The classification accuracy with constraint of semantic keywords.

The classification accuracy is not high in Fig. 4. By observing the training and testing sets, we found that there are not enough training objects in some classes, and some training/testing objects have few extracted terms because of they are non-text pages or link-only pages. Thus, another criteria are used to evaluate the classification accuracy. The testing objects are limited in the class whose number of *semantic keywords*[7] is not less than $CK\#$, where $CK\#$ is an integer. We simulate $CK\#$ from 10 to 50 with interval 10. The result shown in Fig. 5 indicates that the more semantic keywords, the higher classification accuracy. It hsows if the training set is large enough, ACIRD can perform auto-classification with acceptable accuracy.

5. Conclusions

In this extended abstract, we have introduced the design of ACIRD. From the experiment, we conclude that data mining technique is capable of learning the classification knowledge for the Internet document auto-classification.

[7] Based on the recall and precision shown in Table 1, $MG \geq 0.5$ is the best threshold to filter out class keywords. Thus, only keywords with $MG \geq 0.5$ are choosed as keywords in $Know_c^*$, and we call the keyword as *semantic keyword*.

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," IP&M, Vol. 4, 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", to appear in Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98), Melbourne, Australia, August 24-28, 1998.

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

[9] G. Salton, "Automatic Text Processing", 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), Taipei, 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] S. H. Lin, C. S. Shih, M. C. Chen, J. M. Ho, M. T. Kao, and Y. M. Huang, "An Intelligent Internet Information System Based on Classification Knowledge", submitted to IEEE Transactions on Knowledge and Data Engineering.

[15] M. C. Chen and J. M. Ho, "An Automatic Classifier and Explore for Internet Resource", NSC Technical Report, Taiwan, 1997.

[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.