Extracting Classification Knowledge of Internet Documents with Mining Term Associations: A Semantic Approach

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*

Abstract In this paper, we present a system that extracts and generalizes terms from Internet documents to represent classification knowledge of a given class hierarchy. We propose a measurement to evaluate the importance of a term with respect to a class in the class hierarchy, and denote it as *support*. With a given threshold, terms with high supports are sifted as *keywords* of a class, and terms with low supports are filtered out. To further enhance the recall of this approach, *Mining Association Rules* technique is applied to mine the association between terms. An inference model is composed of these association relations and the previously computed supports of the terms in the class. To increase the recall rate of the keyword selection process, we then present a polynomial-time inference algorithm to promote a term, strongly associated to a known keyword, to a keyword. According to our experiment results on the collected Internet documents from Yam search engine, we show that the proposed methods in the paper contribute to refine the classification knowledge and increase the recall of keyword selection.

1. Introduction

Recently, Information Retrieval (IR) systems have become popular due to plentiful information services available on the Internet. The IR systems are designed to facilitate rapid retrieval of information for diverse users using Internet browsers. By applying keyword-based index/search approaches, terms[1] in the document are extracted, indexed and stored in the database. Then the index is employed to rapidly retrieve relevant documents matched the terms in a query. However, the conceptual gap between document authors and the Internet users makes the retrieval results from many conventional IR systems far from user's information needs so that both recall and precision are low. Due to various usage of words, authors and users frequently represent the same semantic with different terms, or describe different meanings by the same word. Therefore, in IR systems, positive term-matched documents are probably wrong answers, such as the term apple has different meanings in "Apple computer" and "apple pie". Therefore, many Internet keyword-based search engines usually retrieve hundreds of documents with the desired few. In contrast, negative matches may be right. For instance, the terms "airline schedule", which does not match with query term "flight schedule". However, both terms have the same semantics. A thesaurus database of terms for a specific domain is probable able to handle the problem. However, because of diverse topics of documents with different culture background on the Internet, no static thesauri can cover the shifting semantics to deal with the term mismatch problem.

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[2] submitted to a search engine implemented with VSM (vector space model) [3], thousands of documents are probably retrieved for the query. In addition, ranking of so many documents according to one or two words is not always meaningful. Therefore, Internet IR systems should be able to efficiently shrink the size of retrieval and to enhance the semantic of documents ranking. Assigning classifications to documents is essential to the efficient management and retrieval of knowledge [2]. In our system, for every class, the classification knowledge is learned from the training data, which are Internet documents collected and manually classified by Yam[3]. By *mining* and *inferring term associations* in each class, the hidden semantics of terms are discovered. With the classification knowledge, a *two-phase search engine* is proposed that performs class-level search first, and then object-level search. In this way, a linear ranking model of documents used in conventional IR systems becomes a structured ranking model.

In the rest of the paper, we discuss related work of IR and mining association rules in Section 2. In order to illustrate our ideas clearly, our system, Automatic Classifier for Internet Resource Discovery (ACIRD), is described in Section 3. In Section 4 and Section 5, we present the details of mining term associations and applying associations to efficiently refine the semantics of terms in a class. From the experiments on the actual categorization of Yam in Section 6, we found that the proposed approaches meet the claimed goal. Finally, we conclude the contributions and point out related future work.

2. Related Work

The past studies in IR systems improved retrieval efficiency by applying techniques of index and query formulation. According to index methods of documents, we can categorize IR systems into two types of systems.

[1] In the paper, we use "term" as word or phrase, which is extracted from the Internet documents, rather than use "keyword". "Keyword" is representative to the concept, while "term" is probably non-representative.

This work was supported pa in part by NSC grant NSC 87-2213-E-001-003.

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

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

Full-text search systems

Full-text search technology has been used in Information Retrieval for many years, which is applied to find matched strings and sub-strings in the documents. Some string-index technologies, such as PAT-tree [11], 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 searching, and regular expression searching [10]. These searching functions are rarely used on the Internet environment; the improvement is seldom used in the Internet. As a novel application, PAT-tree can be used to discover new Chinese terms such as names or terminology [12]. The main disadvantage of full-text search is its inefficient retrieval. Another drawback is low recall and precision rate, since this approach only finds documents with exact matched strings that is usually far from user's information needs.

Keyword-base search systems

Keyword-based search systems usually extract terms from documents based on a pre-constructed dictionary, and index the terms in inverted file [13]. The extracted terms are regarded as keywords and used to represent the document. When a query consisting of one or more terms is submitted to the system, the index is applied to rapidly locate documents that contain these terms. Once documents are retrieved, a VSM-based [3] similarity measurement is employed to rank the documents. A popular ranking algorithm, TF×IDF [14], combines *Term Frequency* and *Inverse Document Frequency* to estimate the metric of relevance between the terms of documents and the query. Those systems can search relevant documents efficiently. However, the index size is usually larger than the size of original documents. Huge physical memory (up to several giga bytes) is required to retain good performance. Otherwise, the page faults of operating system will dramatically degrade the searching performance. Another disadvantage of the systems is that queries with non-indexed terms only can not find any documents. Thus, a complete dictionary with the capability of automatically discovering new terms is critical for keyword-based systems. For the example of Chinese IR environment, the terminology and names are formed by more than one Chinese character and most of them do not exist in the dictionary. An automatic identification of the names is demanded.

Mining Association Rules

Mining association rules [5,6,7] is applied to discover the important associations among items in transactions. A well-known application of item association mining is to find an optimal item arrangement in the supermarket to allow customs to collect the needed grocery conveniently. According to the definition of association rule in [6], 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 in D that contains X also contains Y. The rule $X \rightarrow Y$ has support s in the transaction set D if s% of transactions in D that contain $X \cup Y$.

We follow the above definition and map the problem of mining term associations to the specification. The detail of mining term associations is described in Section 4.

3. The ACIRD System

The system, Automatic Classifier for Internet Resource Discovery (**ACIRD**[4]) [9], is implemented to automatically classify documents collected by Yam Web Server. The system is motivated to improve the poor performance of the current manual classification. The classification process of ACIRD is composed of two phases, *training phase* and *testing phase*. In the first phase, documents with their manually assigned classes in Yam are employed as the training set to learn the classification knowledge of the classes in the lattice. Then the newly collected documents manually categorized by Yam's experts are applied to verify the learned classification knowledge in the second phase.

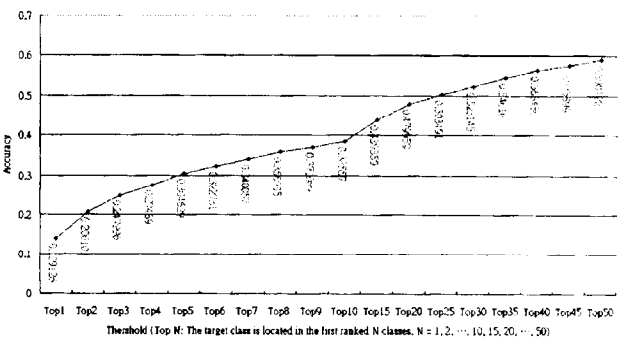

Figure 1: The Classification Accuracy of ACIRD (Exact Match).

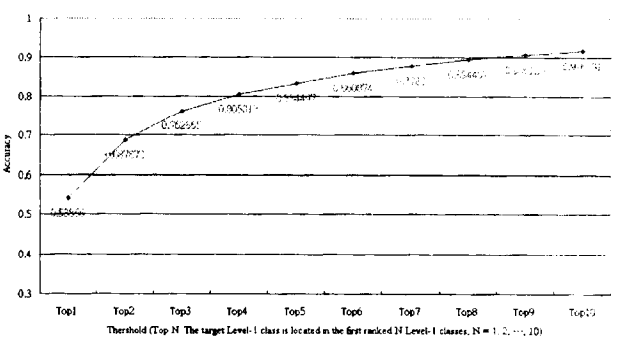

Figure 2: The Classification Accuracy of ACIRD (Level-1 Classification).

Results of the current training and testing phase is shown in Figure 1. Given a document with a target class among 513 classes of Yam, the accuracy of exact match (Top1) is 13.2%. The probability that the target class appears in the first 5 classes (Top5) among 513 classes is about 30.5%. If we apply the same classification process to classify Level-1 classes of

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

[5] There are 12 main categories in Yam's homepage (http://taiwan.iis.sinica.edu.tw/en/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".

Yam[6], the result is shown in Figure 2. Among 12 Level-1 classes, the accuracy of exact match (Top1) is 53.9%, and the probability of accurate classification to the half of Level-1 classes (Top6) is about 86.1%.

Currently, the automatic classifier is used as assistance to Yam's colleagues to speed up the performance of categorization process. *Learning by advice* is also applied to refine the classification knowledge during the period of manual interactions. In the following, we describe the system based on the issues of IR systems [10]: *document operation*, *conceptual model*, *term operation*, *file structure*, *query operation*, and *hardware*. The following terminology is used throughout the paper.

Class corresponds to a Yam's category. The classification knowledge of a class is represented by a group of keywords.

Object corresponds to an Internet document (Web page).

Term is the word or phrase extracted from objects or generalized into classes by the learning process.

Support is the improtance degree of a term that supports some object or class. The value is normalized to [0, 1].

Keyword corresponds a representative term, i.e., its information quality is better than a term.

Memership grade (MG) is the supporting degree of a keyword to some object or class as if support is the supporting degree of a term.

Document Operation

Each document (object) is assigned a unique OID (object ID) in the database. An object can be instantiated from several classes according to Yam's current categorization. Before performing the classification process, objects are preprocessed into terms with supports $sup_{t_i \to O}$, t_i is a term extracted from object O.

Web document authors highlight the content with HTML tags, such as title, heading, list, table, italic, bold, underline, etc. Additionally, META tag allows users to add extra information for the documents such as "classification" and "keyword". Apparently, HTML tags provide significant information to index and classification. We categorize HTML Tag into four types:

1. *Informative*: Contents in these tags are regarded most important, such as <META Classification Keyword>, <TITLE>, <H1-6>, <B>, <I>, etc. Thus, the weight of content enclosed by these tags is increased.
2. *Skippable*: These tags have no effects, such as
, <P>, etc. These tags are omitted during processing.
3. *Uninformative*: Contents enclosed by these tags, such as <AREA>, <COL>, <COMMENT>, etc., are not processed and indexed.
4. *Statistical*: Contents included in these tags, such as <!DOCTPYE>, <APPLET>, <OBJECT>, <SCRIPT>, etc., are extracted and stored in database as statistical information in the future.

The content is usually included in nested tags (called "TagPattern"), and only the tag with maximal weight in TagPattern is considered as the weight of the content. Both weight and frequency are used to measure $sup_{t_i,O}$ of a term extracted from the object. With normalization, the measure of $sup_{t_i,O}$ is shown in (3.1).

$$sup'_{t_i,O} = \sum_{T_j} tf_{ij} \cdot w_{T_j}, \quad (3.1)$$

t_i is a term enclosed by TagPattern T_j in O,

tf_{ij} is the term frequency in T_j, and

w_{T_j} is the maximal weighed tag in T_j.

$$sup_{t_i,O} = \frac{sup'_{t_i \to O}}{\underset{t_i \text{ in } O}{MAX}(sup'_{t_i \to O})}$$

Conceptual Model

Probabilistic document conceptual model is applied to the system. For each object, terms are extracted from its context. Objects categorized by Yam are the training set of the classification process. By learning terms from the training set of a class, terms with supports to the class are used to represent the classification knowledge of the class. The support of a term to a class C is measured by (3.2).

$$S_{t_i,C} = \frac{\sum_j sup_{t_i \to O_j^i}}{\underset{t_i \text{ in } C}{MAX}(\sum_j sup_{t_i \to O_j^i})}, \quad t_i \text{ is a term in } C, \quad (3.2)$$

and an object O_j in C containing t_i is donoted as O_j^i.

For $S_{t_i,C}$, a threshold θ is defined to divide terms into two types.

Representative: a term t_i is representative, if $S_{t_i,C} \geq \theta$.

Non-representative: a term t_i is non - representative, if $S_{t_i,C} < \theta$.

For each class, terms are applied to mine associations between terms, which is named *term associations*. Based on the digraph constructed from term associations and term supports, an inference model of terms to the class is formed to promote some terms from non-representative to representative states. A term t_i becomes a keyword k_i of a class, if representative. $MG_{k_i,C}$ is used to substitute $S_{t_i,C}$, if t_i is promoted to k_i. Thus, $MG_{k_i,C}$ is the maximal value between $S_{t_i,C}$ and the value inferred from term associations. The detail of the inference model is described in Section 5.

In query model, a query is formulated with terms concatenated with Boolean operators (AND, OR, NOT). Similarity match based on VSM is applied to two-phase search (introduced in the following *query operation*) to retrieve hierarchical results.

Term Operation

Terms in an object are extracted according to our developed Term-Segmentation Rules based on the term database. The database is mainly originated from the analysis of past query logs of Yam. Besides, during the periods of automatic classification and mining term associations, the system also constructs *term semantic networks* (*TSN*) as the system's thesaurus.

File Structure

In the system, keyword-based inverted indexes of keywords (or terms) in classes and objects are implemented by the relational database system, SQL Server 6.5. Given a term or a

[6] There are 12 main categories in Yam's homepage (http://taiwan.iis.sinica.edu.tw/en/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".

keyword, relevant classes (indicated with CID) or objects (indicated with OID) can be retrieved efficiently. IN the experiment, there are 37986 indexed terms associated to 8706 indexed objects and 512 indexed classes. In average, given a query term, the system costs 1.132 ms to retrieve relevant objects, and 0.041 ms to retrieve relevant classes.

Query Operation

By applying refined classification knowledge (keywords in the class), two-phase search is used in the system. In the first phase, query terms are used to search qualified classes. Those classes form a shrunk view of Yam's hierarchical lattice to shrink the searching domain of the following object search. In the second phase, for each matched class, query terms (except that terms are matched with the class) are employed to match objects in the class. By integrating qualified classes and matched objects in these class, a tree (or lattice) with probabilities on branches is shown to the user. Thus, the two-phase search mechanism shrinks the searching domain and presents a hierarchical view to users.

Hardware

Currently, the system is implemented on a Pentium II 233/128M SDRAM machine with NT Server 4.0 (SP3) and SQL Server 6.5 (SP1) software systems.

In Fig. 3, we summarize the overview of the system. First, *Document Collection* and *Class Lattice* are borrowed from Yam inventory. Then, terms with supports are extracted from documents and are further generalized as *Classification Knowledge*. Finally, the classification knowledge is refined to a set of keywords (i.e., *Refined Classification Knowledge*) by employing mining *Term Association Rules* in each class. For query processing, the two-phase search module parses *Query* into term-based *Query Representation* that is used to match with *Refined Classification Knowledge* and documents indexed by terms. The semantics of synonym or alias have included the knowledge base.

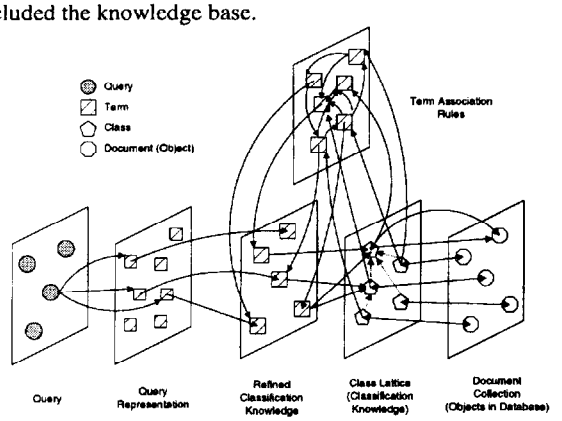

Figure 3: Overview of the Process Flow of ACIRD

4. Mining Term Associations

After generalizing terms of the class, these terms form a graph with direct edges from term vertices to the class vertex. The edge is labeled with support $S_{t_i,C}$ of term t_i to the class C. Thus, we call the graph as Term Support Graph (**TSG**). In average, there are only a few representative terms (e.g., $S_{t_i,C} \geq 0.5$) for a class, therefore, the recall of the class knowledge is low. Thus, mining the hidden semantics among terms can alleviate the problem. In our system, we apply *Mining Association Rules* to discover those excluded but representative terms. Based on mining associations from those terms, the system constructs Term Association Graph (**TAG**). Combining TAG and TSG, Term Semantic Network (**TSN**) is constructed as shown in Figure 4. In this section, we describe the process of mining term associations. In next section, the algorithm of inference on TSN is introduced.

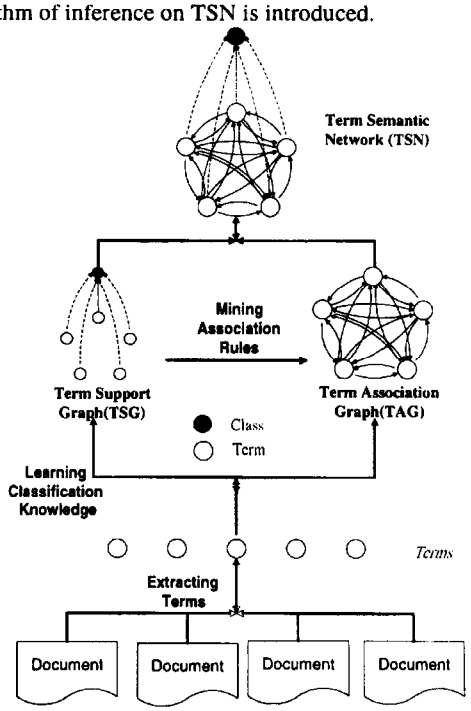

Figure 4: Construction of Term Semantic Network

Two important issues should be considered before mining term associations is described. One is what granularity should be used to mine associations. The other is what boundary should be to generate association rules, which is the *transaction database* defined in [6].

The granularity of mining associations

In [8], authors restricted the granularity of generating associations to 3-10 sentences per paragraph. It is infeasible since a paragraph may be hundreds of sentences, the system should not skip later sentences. Additionally, the importance of a sentence of Internet document depends on HTML tags (HTML weights defined in the system). Therefore, we can regard an object's *informative paragraphs* as a transaction defined in [6].

The boundary of generating association rules

As Internet documents are published by a diversity of people, the semantics of words alternate case by case. It is possible to use the same word for different concepts, which arise from various cultures of Internet users. For example, if a query with "apple" and "computer", the semantics can not be "apples of food" (As we know, it means "Macintosh"). Thus, in the system, we apply mining term associations based on documents of each class rather than all documents in the database. Based on those reasons, we translate our problem domain into the domain of mining association rules [6].

Terms are corresponding to items.

Documents in the class are corresponding to transactions.

The class is corresponding to the transaction database.

Concentrating on documents of a class instead of all classes also takes the advantage of small database size, as the complexity of mining association problem is exponentially increased by the size of transaction database. If the size of database is not very large, a simple mining algorithm, such as Apriori [5], can be efficiently applied to our system.

As for the *confidence* and *support* defined in [6], the former is used to promote non-representative term to refine the classification knowledge, and the latter is recognized as a threshold to filter out noisy term associations. For example, after mining term associations in class "Art", the confidence and support of "exhibition" → "art" are 0.826 and 0.1. By experience, an association rule with 10% supports is useful. The association can be applied to refine the classification knowledge. In the classification knowledge, $S_{t,C}$ of term "exhibition" to class "Art" is 0.13, and $S_{t,C}$ of term "art" to class "Art" is 1. According to the inference of TSN, introduced in Section 5, in the class "Art", $S_{t,C}$ of term "exhibition" to class "Art" is promoted from 0.13 to 0.826 (The promoted $S_{t,C}$ of the term "exhibition" = 0.826*1 = 0.826.).

Based on the original support, the term "exhibition" may be filtered out due to its low support (0.13). However, the promoted value (0.826) is enough to support the term "exhibition" as a keyword of the class "Art". Thus, the classification knowledge is refined.

Moreover, by storing the information of term-location in a document, new Chinese terms (or *compound terms*) can be easily discovered. For example, in the class "Computer Hardware", 麥 ←→ 金 (A ←→ B represents "if and only if"), 麥 ←→ 塔, and 金 ←→ 塔 are with (*confidence*, *support*) = (1, 0.25). After retrieving their term-locations from the database, the system detects that all documents satisfy "麥 is followed by 金, and 金 is followed by 塔". Thus, we discover that "麥金塔 (Macintosh)" should be a new Chinese term used in the class "Computer Hardware". A new terminology is learned in this way.

In our system, all term associations in a class are automatically applied to refine the classification knowledge of corresponding classes (in Section 5). In this way, no query expansion process is necessary. As we mentioned in previous example, a query with the term "exhibition" does not need to be expanded as "exhibition" and "Art", since "exhibition" has been included in the class "Art" by promoting its support to the class. Also, by directly applying term associations to refining knowledge (instead of expanding queries), the system is never confused by which domain (class) thesaurus should be employed. Moreover, the performance will be deteriorated that the query expansion should be performed every time to match with each class, if different classes own their domain thesaurus.

5. Optimization of Term Semantic Network

We want to promote some non-representative terms to *promoted-representative* after mining term associations and constructing TSN. In other words, the term's support is larger than the threshold after the optimization process. The optimization process of TSN is shown in Figure 5. In the left-hand side graph (TSN), there are four *non-representative* terms. In the right-hand side graph (Optimized TSN), three terms are promoted to the *promoted-representative* state based on their associations with the *representative* term. The optimization of TSN is based on the inference chain in the graph. As we can see in the figure, three terms with *supports* are promoted to keywords with *MG* after optimizing TSN. Before describing how to optimize TSN, we first give a formal definition of TSN.

Graph Notations

Referring the graphic notations using in [15], for a graph G, the vertex set of G is denoted as $V(G)$ and the edge set of G is denoted as $E(G)$. A path in G is an alternative sequence of vertices and edges, beginning and ending with vertices.

$p: v_0, e_1, v_1, e_2, v_2, \ldots, v_{n-1}, e_n, v_n (n \geq 0)$, $e_i = v_{i-1}v_i$, $i = 1,2,\ldots,n$.

A path from vertex u to vertex v is denoted as $p(u,v)$ and the set of all possible paths is denoted as $P(u,v)$.

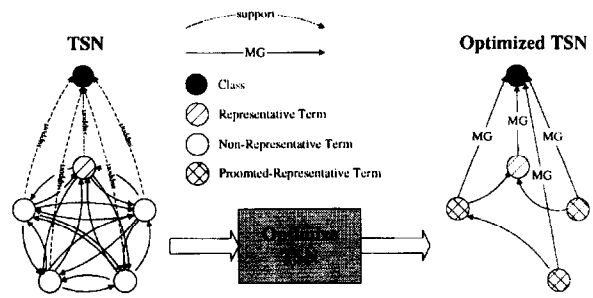

Figure 5: Optimization of TSN

Term Association Graph (TAG)

TAG, a strongly connected digraph, describes the association among terms. Each vertex u in $V(TAG)$ represents a term, and each edge uv in $E(TAG)$ denotes the mined association rule "$u \rightarrow v$" with confidence $conf_{uv}$ as the edge's label.

$conf_{uv} = \frac{f(u \cap v)}{f(u)}$, *where* $f(u)$ stands for the number of documents that contain term u.

The association between $u \rightarrow v$ is asymmetric.

A path (u,v,w) concatenated by edge uv and edge vw forms the inference chain that indicates the probability of co-existence of these three terms. Based on the assumption of conditional independence, the combination rule to reason $conf_{uvw}$ is defined below.

$0 \leq conf_{uvw} = conf_{uv} \cdot conf_{vw} \leq 1$,

where $u, v, w \in V(TAG)$, and $uv, vw, uw \in E(TAG)$

Term Support Graph (TSG)

TSG is a digraph in which one class-vertex C represents the class, and a set of term vertices u that also exists in TAG denote these terms supporting the class. Only one type of edge, from a term u to the class C, is used to represent the term's support to C. Each edge is labeled with the support S_{uC} $(0 \leq S_{uC} \leq 1)$ as defined in Section 3.

Term Semantic Network (TSN)

TSN is the union of TAG and TSG as shown in Figure 4.

$V(TSN) = V(TSG) \cup V(TAG) = V(TSG)$

$E(TSN) = E(TSG) \cup E(TAG)$

As we described previously, only a few terms in TSN are representative, many non-representative terms should be promoted by the *association-chain* from these non-representative terms to representative terms. Through the association-chain, a term has many *alternative-supports* that can be use to decide *term promotion*.

Alternative support

Association-chain is a path started from a term vertex u via the other term vertices $v_1, v_2, ..., v_n$, and finally ended in C. The inference on the chain denotes an *alternative support* of u to C.

$0 \le as_{uv_1v_2...v_nC} = conf_{uv_1} \cdot conf_{v_1v_2} \cdots conf_{v_{n-1}v_n} \cdot S_{v_nC} \le 1,$

where $u, v_i \in V(TAG), C \in V(TSG)$

For any term vertex u, there are many possible paths (*alternative supports*) to C. The set of these supports is denoted as AS_{uC}. While optimizing TSN, we want to find a term's maximal measure among S_{uC} and AS_{uC}. The maximal measure is regarded as *MG* of the *keyword* (if the term is promoted) to the class.

Membership Grade (MG)

$MG_{uC} = MAX(S_{uC}, AS_{uC})$, if $MAX(S_{uC}, AS_{uC}) \ge threshold$

else $MG_{uC} = 0$, i.e., u is not a keyword of C after the promotion

Considering TSN with n terms, since TAG is strongly connected, there are $n \cdot \sum_{i=1}^{n-1} P_i^{n-1}$ alternative paths[7]. Suppose a class has 10 terms, there are 2.3×10^8 possible paths. Unfortunately, after leaning the classification knowledge, the average number of terms of a class is 43. It's the problem of combination explosion, and the exhaustive search is not possible. Thus, based on greedy approach, we propose Perfect Term Support (PTS) algorithm to find each term's *MG* in polynomial time.

Perfect Term Support (PTS) Algorithm

Since *MG* of u is measured by determining the maximum among S_{uC} and AS_{uC} (which is equal to $conf_{u...v} \cdot S_{vC}$) of u, *MG* is fully dependent on $S_{vC}, v \in V(TSN)$. Based on the greedy heuristic, the first selected term is the one with maximal $S_{uC}, u \in V(TSN)$, and S_{uC} is just the term's *MG*. Theorem 1 proves $MG_{uC} = S_{uC}$, if u is the term with maximal S_{uC}.

Theorem 1:

In TSN, if u exists such that $S_{uC} = MAX(S_{vC})$,

where $u, v \in V(TSN)$ and $vC \in E(TSN)$, then $MG_{uC} = S_{uC}$

Proof:

For a path $(u, v_1, \cdots, v_n, C) \in P(u, C)$, $S_{uv_1 \cdots v_nC} = conf_{uv_1 \cdots v_n} \cdot S_{v_nC}$

$\because S_{uC} = MAX(S_{vC})$, for $\forall v \in V(TSN)$ and $vC \in E(TSN)$,

i.e., $S_{uC} > S_{vC}$ where $v \ne u$, $vC \in E(KSN) - uC$

$$\frac{S_{uv_1 \cdots v_nC}}{S_{uC}} = \frac{conf_{vv_1 \cdots v_n} \cdot S_{v_nC}}{S_{uC}} = \frac{S_{v_nC}}{S_{uC}} \cdot conf_{vv_1 \cdots v_n} < conf_{vv_1 \cdots v_n} \le 1$$

Therefore, we say $MG_{uC} = S_{uC}$.

Based on Theorem 1, PTS algorithm is a greedy algorithm that guarantees *MG* of each term is founded after the algorithm terminates. Starting from the class vertex C in TSN, PTS selects the vertex, called u_1, with the maximal support S_{uC} to vertex C in the initial state. Then u_1 is added into the set termed Set_{MG}. PTS guarantees the term's support is *MG* to the class while it is added into Set_{MG}. In the following loops, through u_1 to C, alternative supports of remaining terms are estimated. For each term, if the maximal alternative support is larger than its original support, the support is replaced by the maximal alternative one. Among all remaining terms, the term vertex with maximal (*updated*) support is added into Set_{MG}. The algorithm is terminated while $Set_{MG} = V(TSN) - C$, and *MG* of each term is found.

PTS Algorithm:

1. *Initial state*: [This step sets the labels ($\ell(u)$, the current maximal support of u) of all term vertices in TSN to S_{uC}, i.e., $\ell(u) = S_{uC}$. Vertex u_1 with the maximal support to C is added into Set_{MG} that contains vertices of TSN whose *MG* to C is determined up to now. The set or remaining terms, $\overline{Set_{MG}}$, is initialized as the set $V(TSN) - \{C, u_1\}$.]

Let the loop index $i \leftarrow 1$; $\ell(u_1) \leftarrow MAX(S_{vC})$;

$Set_{MG} \leftarrow u_1$; $\overline{Set_{MG}} \leftarrow V(TSN) - \{C, u_1\}$;

$\ell(u) \leftarrow S_{uC}$ for all $u \in V(TSN) - u_1$;

If $\overline{Set_{MG}}$ is empty (i.e., $V(TSN) = \{C, u_1\}$) then stop;

otherwise, continue;

2. [For each vertex u in $\overline{Set_{MG}}$, this step updates $\ell(u)$, if necessary, on account of the path from u to C via u_i in Set_{MG}. Further, if $\ell(u)$ is changed, then $ANTECEDENT(u)$[8] is assigned to vertex u_i to keep the previous vertex of u.]

For each $u \in \overline{Set_{MG}}$ such that $uu_i \in E(TSN)$,

If $conf_{uu_i} \cdot \ell(u_i) > \ell(u)$,

then $\ell(u) \leftarrow conf_{uu_i} \cdot \ell(u_i)$ and $ANTECEDENT(u) \leftarrow u_i$;

3. [This step finds the vertex with $MAX(\ell(u))$ from $\overline{Set_{MG}}$, and set the vertex as u_{i+1}.]

Determine $m = MAX\{\ell(v) \mid v \in \overline{Set_{MG}}\}$;

If $v_j \in \overline{Set_{MG}}$ is selected as a vertex with $\ell(v_j) = m$,

then output m as MG of v_j to C and $u_{i+1} \leftarrow v_j$;

4. [This step adds u_{i+1} into Set_{MG} and removes it from $\overline{Set_{MG}}$.]

---

[7] $n \cdot \left[(n-1)! + \frac{(n-1)!}{1!} + \frac{(n-1)!}{2!} + ... + \frac{(n-1)!}{(n-2)!} \right]$

$= n \cdot \sum_{i=1}^{n-1} P_i^{n-1}$, where $P_i^{n-1} = \frac{(n-1)!}{(n-1-i)!}$

[8] By tracing u's antecedent vertex recursively, PTS can identify vertices of the path that supports u's *MG*.

$Set_{MG} \leftarrow Set_{MG} \cup u_{i+1}$ and $\overline{Set_{MG}} \leftarrow \overline{Set_{MG}} - u_{i+1}$;

5. [This step increases the loop index and determines whether the algorithm finishes.]

$i \leftarrow i+1$; If $\overline{Set_{MG}}$ is empty, then stop; otherwise, go to step 2;

An Example of PTS

As an example, we apply PTS algorithm to optimize TSN of the class "Placard & Photography" shown in Figure 6. Table 1 indicates initial supports of terms to the class. The first column indicates supports of terms, and the other columns show the term associations.

Table 2 has columns headed from u_1 to u_5, one for each term. The ordered pair in the column corresponding to the vertex u_i indicates $(\ell(u_i), ANTECEDENT(u_i))$ for $i=1,2,3,4,5$. The last row compares the initial support and the promoted support. With term associations, most of the supports are promoted in the case. For example, the support of u_4 raises from 0.10 to 0.83 because of the high term association from u_4 to u_1.

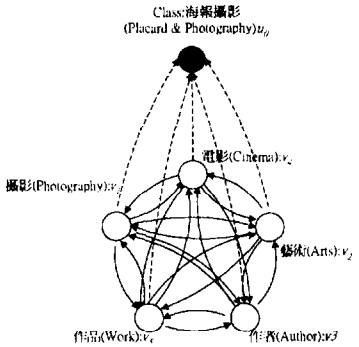

Figure 6: TSN of Class " Placard & Photography".

Table 1: Initial Supports of Terms in Class "Placard & Photography".

Placard & Photography (C)		Photo-graphy (u_1)	Arts (u_2)	Author (u_3)	Cinema (u_4)	Work (u_5)
1.00	Photo. (u_1)	-	0.06	0.20	0.33	0.47
0.13	Arts (u_2)	0.25	-	0.25	0.25	0.5
0.11	Author (u_3)	0.75	0.25	-	0.50	0.25
0.10	Cinema (u_4)	0.83	0.20	0.33	-	0.67
0.19	Work (u_5)	0.78	0.22	0.11	0.44	-

Table 2: Loops of PTS for Class "Placard & Photography".

	u_1	u_2	u_3	u_4	u_5	Set_{MG}
i=1	$(1.0, C)$	$(0.13, C)$	$(0.11, C)$	$(0.10, C)$	$(0.19, C)$	u_1
i=2		$(0.25, u_1)$	$(0.75, u_1)$	$(0.83, u_1)$	$(0.78, u_1)$	$\{u_1, u_5\}$
i=3		$(0.39, u_5)$	$(0.75, u_1)$	$(0.83, u_1)$		$\{u_1, u_5, u_4\}$
i=4		$(0.39, u_5)$	$(0.75, u_1)$			$\{u_1, u_5, u_4, u_3\}$
i=5		$(0.39, u_5)$				$\{u_1, u_5, u_4, u_3, u_2\}$
Initial/ Promoted	1.0/1.0	0.13/0.39	0.11/0.75	0.10/0.83	0.19/0.78	

Using Table 2, we can obtain the following paths that MG of terms are optimized with maximal supports:

$p_1 : u_1, C;$

$p_2 : u_2, u_5, u_1, C;$

$p_3 : u_3, u_1, C;$

$p_4 : u_4, u_1, C;$

$p_5 : u_5, u_1, C;$

Completeness of PTS Algorithm

Now, we verify the validity of PTS algorithm. Theorem 2 describes that as PTS terminates MG of each term vertex is found. The proof of Theorem 2 is given as the validity of the algorithm.

Theorem 2:

When the algorithm terminates, (5.1)

$MG_{uC} = \ell(u)$ for all $u \in V(TSN) - C$.

Further, if $\ell(u) \neq 0$ and $u \neq C$,

then $u = w_k, \cdots, w_2, w_1, w_0 = C$ is a path (5.2)

with maximum support (i.e., MG),

where $w_{i-1} = ANTECEDENT\ (w_i)$ for $i = 1,2,\ldots, k$

Proof:

We begin with verifying (5.1).

We prove based on induction of the loop index (i, where $1 \le i \le N-1$ and $N = |V(TSN)-C|$) and show that, after u_i is determined,

$\ell(u) = MG_{uC}$ for all $u \in Set^i_{MG} = \{u_1, \cdots, u_i\}$, (5.3)

where i indicates Set_{MG} in ith loop.

This is certainly true for $i=1$, based on Theorem 1.

$\ell^1(u_1) = MAX\ (S_{uC})$, where $\ell^1(v)$ stands for the current maximal support in ith loop.

According to Theorem 1, because $\ell^1(u_1)$ is the maximum support of edge vC for all $v \in \overline{Set^1_{MG}}$ where $\overline{Set^1_{MG}} = V(TSN) - C$, therefore $MG_{u_1C} = \ell^1(u_1) = S_{u_1C}$.

Assume Theorem 2 is true for $i = N-1$. u_{N-1} is the vertex such that $\ell(u_{N-1}) = \ell^{N-1}(u_{N-1}) = MAX\ \{\ell^{N-1}(v) \mid v \in \overline{Set^{N-1}_{MG}}\}$ and $MG_{u_{N-1}C} = \ell(u_{N-1})$.

And for all $v \in \overline{Set^{N-1}_{MG}} - u_{N-1}$, we update $\ell^{N-1}(v) = MAX(\{conf_{vw} \cdot MG_{wC} \mid w \in Set^{N-1}_{MG}\}, S_{vC})$

In other words, $\ell^{N-1}(u_{N-1})$ is the maximal value among $\ell^{N-1}(v)$, where $v \in \overline{Set^{N-1}_{MG}}$ when $i = N-1$.

Then we prove the theorem is true when $i = N$.

For each $v \in \overline{Set^N_{MG}} = \overline{Set^{N-1}_{MG}} - u_{N-1}$, by PTS, we can obtain $\ell^N(v) = MAX(\ell^{N-1}(v), conf_{vu_{N-1}} \cdot MG_{u_{N-1}C})$.

Based on the assumption of induction, $\ell^{N-1}(u_{N-1})$ is the maximum of $\ell^{N-1}(w), w \in \overline{Set^{N-1}_{MG}}$, and $\ell^N(v)$ is the maximum of $\ell^{N-1}(v)$ and $conf_{vu_{N-1}} \cdot MG_{u_{N-1}C}$, we can say $\ell^N(v)$ is the maximal support of edge vC and all alternative paths via $w \in Set^N_{MG}$.

By PTS, u_N is a vertex such that $\ell(u_N) = MAX\{\ell(v) \mid v \in \overline{Set^N_{MG}}\}$.

Because the alternative support of the path $(u_N, w, ...C)$ where $w \in Set^N_{MG}$ is larger than that of any other path $(v, w, ...C)$ where $w \in Set^N_{MG}$ and $v \in \overline{Set^N_{MG}} - u_N$, by definition of MG, we can say

$MG_{u_N C} = MAX(\{conf_{uNv} \cdot MG_{vC} \mid v \in Set^N_{MG}\}, S_{u_N C})$.

$= \ell^N(u_N)$

By the induction hypothesis, we can say when PTS terminates,

$MG_{vC} = \ell(v)$, for all $v \in V(KSN) - C$.

To verify (5.2), let $v \in V(KSN)$ such that $\ell(v) \neq 0$, and $v \neq C$. At the completion of the algorithm,

if $ANTECEDENT(v) = C$, (5.2) is certainly true; otherwise

$\ell(v) = conf_{v v_1} \cdot \ell(v_1)$,

where $ANTECEDENT(v) = v_1$ and

$MG_{vC} = conf_{v v_1} \cdot MG_{v_1 C}$.

The fact implies that v_1 is the second to last vertex on some (v, C) path with MG.

Continuing in this manner, we produce a (v, C) path with MG.

$P: v, v_1, \cdots, v_{n-1}, v_n = C$, where $ANTECEDENT(v_i) = v_{i+1}$ for $i = 1, 2, ..., n-1$. Let $w_i = v_{n-i}$ for $i = 0, 1, ..., n$. Then we see that (5.2) follows.

As observing PTS algorithm, the time complexity is $O(n^2)$ which provides a better performance comparing with the exhaustive inference mechanism whose time complexity is exponential.

6. Experiment Results

We examined the performance of our algorithms using the collected data from YAM. Experiments are designed to verify if ACIRD can learn and promote representative terms (keywords), which should approach to concepts of human experts for each class. Thus, recall and precision of the keywords extracted by ACIRD are compared with the keywords manually selected by experts. Four criteria are used to qualify class keywords generalized and promoted by PTS.

- **Top 10**: the membership grades of all terms are sorted first. The first 10 ranked terms are selected to be the class keywords.
- **Top 20**: the first 20 ranked terms from the sorted terms are selected to be the class keywords.
- **Threshold = 0.5**: this criterion selects tho terms whose *MG* is larger than 0.5.
- **Threshold = 0.7**: this criterion selects tho terms whose *MG* is larger than 0.7.

Experiments are tested for the precisoin and recall of classes keywords before and after the PTS algorithm is applied respectively. Experiment results are shown in Table 3. Observing the results, before applying PTS, we can find that the precision of all criteria is larger than 0.73 no matter TSN is optimized or not. Based on these high thresholds (Top 10, Top 20, T = 0.5, and T = 0.7), most of qualified terms are *representative*, thus the precision is high. Such result also supports that our classification learning approach can generalize representative terms based on these thresholds. However, as we predicted, the recall is very low. It implies that the learning approach can not learn the implicit semantic without PTS. As for PTS, in comparison with the case without applying PTS, the precision is slightly decreased, but the recall is dramatically increased. Thus, by the learning approach, mining term associations and PTS, the system can dig out the hidden semantic (i.e., increase the recall) without losing the precision. Also, in comparison with these criteria, the criterion, "Threshold = 0.5", can be employed in ACIRD based on the evaluation of recall and precision.

Table 3: Simulation Results Based on Precision/Recall.

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

7. Conclusion and Future Work

Based on the experiment, we conclude that mining term associations from the documents is effective to refine the classification knowledge. By the inference of TSN, meaningful terms with low supports, which will be filtered out before PTS is applied, are promoted to enhance recall and precision. In addition, with the term promotion, the meaning of ranking algorithm is much close to the perception of users. Thus, the inference model of TSN indeed represents the semantics of terms.

In the future, we will enhance TSN to model the knowledge representation of thesaurus. In our model, the edge on TSN model only indicates the co-occurrence probability. Observing the constructed term semantic network models, we found that the co-occurrence terms usually have different meanings. Some terms support positive concept but some are negative. With only considering the co-occurrence probability, pro or con of the concept are confused. With the enhancement of these formal semantic relationships in the model, edges can hold specific meanings. According to these specific meanings, we can construct the precise conceptual representation for classes. Also, based on mining term associations and heuristics, new terms with semantic associations can be discovered by the system to improve and enlarge the thesaurus.

8. References

[1] Mostafa, J., Mukhopadhyay, S., Lam, W. and Palakal, M., "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.

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

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

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

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

[6] Agrawal, R., Imielinski, T., 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.

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

[8] Jing, Y. F. and Croft, W. B., "An Association Thesaurus for Information Retrieval", UMass Technical Report 94-17.

[9] Lin, S. H., Chen, M. C., Ho, J. M., and Huang, Y. M., "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.

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

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

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

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

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

[15] Chartrand, G. and Oellermann, O. R., "Applied and Algorithmic Graph Theory", McGraw-Hill, Inc., 1993.