LiveClassifier: Creating Hierarchical Text Classifiers through Web Corpora

Chien-Chung Huang
Institute of Information Science
Academia Sinica
Taipei, Taiwan
villars@iis.sinica.edu.tw

Shui-Lung Chuang
Institute of Information Science
Academia Sinica
Taipei, Taiwan
slchuang@iis.sinica.edu.tw

Lee-Feng Chien *
Institute of Information Science
Academia Sinica
Taipei, Taiwan
lfchien@iis.sinica.edu.tw

ABSTRACT

Many Web information services utilize techniques of information extraction (IE) to collect important facts from the Web. To create more advanced services, one possible method is to discover thematic information from the collected facts through text classification. However, most conventional text classification techniques rely on manual-labelled corpora and are thus ill-suited to cooperate with Web information services with open domains. In this work, we present a system named LiveClassifier that can automatically train classifiers through Web corpora based on user-defined topic hierarchies. Due to its flexibility and convenience, LiveClassifier can be easily adapted for various purposes. New Web information services can be created to fully exploit it; human users can use it to create classifiers for their personal applications. The effectiveness of classifiers created by LiveClassifier is well supported by empirical evidence.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Miscellaneous

General Terms

Algorithms, Experimentation, Performance

Keywords

Text Classification, Web Mining, Topic Hierarchy

1. INTRODUCTION

Although the field of Web information extraction (IE) has made much progress in recent years [7, 6, 12, 22], most of the time the information extracted is simply used to populate the databases without any refining step. If this extracted information can be processed and more information can be discovered from it, undoubtedly, many new advanced information services would become possible. For example, suppose there is a Web information service that helps users collect publication lists of researchers in a certain area; and suppose further that there are some mechanisms to decide the researchers' specialized fields based on their publication lists, such mechanisms would be highly valuable from the perspective of both commercial interests and academic inquisition. In brief, discovering information hidden in the extracted information, e.g., the information at thematic level, would open up possibilities of creating new Web applications and help researchers to conduct deeper analysis.

Generating thematic information can be realized through text classification – a subject having been extensively studied for years [21]. In the above example, one can classify the publication lists into a set of classes representing various fields in computer science, thus determining the interests of researchers. However, most text classification techniques assume manually-labelled corpora are handy and can be used for training – an assumption sometimes not quite realistic in practical experience. For one thing, labelling the corpus is laborious and needs the assistance of professional indexers, and possibly suffers from the problem of subjectivity; for another, the acquisition of corpora is often a non-trivial matter. Therefore, these techniques relying on hand-labelled corpora to create thematic metadata are ill-suited to cooperate with Web information services.

If, on the other hand, there exists a system that can automatically acquire necessary training corpora based on user-defined topic hierarchies and train the classes effectively, positively such a system can be easily adapted to cooperate with Web information services. Moreover, it also would give great facility to human users.

The above consideration motivates us to design a system named *LiveClassifier* that requires limited human involvement in creating hierarchical text classifiers. *LiveClassifier* was developed under the assumption there does not exist any manually-assigned corpus, or even if it exists, the amount is inadequate. Consider that the Web offers inexhaustible data source for almost all subjects, the system was designed to fully exploit the richness of Web resources. The main features of *LiveClassifier* are: (1) using Web search-result pages as the corpus source; (2) exploiting the structural information inherent in the topic hierarchy to train the classifier; and (3) creating key terms to amend the insufficiency of the topic hierarchy. We here sketch the key idea of *LiveClassifier* briefly.

Given a topic hierarchy, an intuitive idea would be to acquire topic-related corpora from the Web to be the substitutes of manual-labelled corpora. However, considering the heterogeneity of Web corpus, much noises must be contained in these downloaded documents. Thankfully, there are more suitable tools to realize this idea—Web search engines. One may send the names of the classes as queries to search engines and use the returned search-results pages as the corpus. However, even so, the retrieved search-results still may contain noises unavoidably. Our idea is that we can formu-

*Lee-Feng Chien also works in Department of Information Management, National Taiwan University, Taiwan

WWW2004, May 17–22, 2004, New York, New York, USA.
ACM 1-58113-844-X/04/0005.

late more precise queries and organize the corpus more effectively by the concept of the classes.

The main merits of *LiveClassifier* are its wide adaptability and its flexibility. The needed classifier can be created by simply defining a topic hierarchy. Aside from generating more thematic information for Web information service, *LiveClassifier* also gives much convenience to human users.

The rest of the paper is structured as follows. *LiveClassifier*, along with the approach it is based on, is presented in Section 2, experiments are presented Section 3, the related work in Section 4, and conclusions are drawn in Section 5.

2. LIVECLASSIFIER

In Section 2.1, we give an overview of the components of *LiveClassifier*. A more detailed analysis of each component is presented respectively in Sections 2.2, 2.3 and 2.4.

2.1 Overview of the System

We first define the problem *LiveClassifier* is supposed to deal with and then discuss the technical details.

Given a set of classes, $C = \{c_1, c_2..., c_n\}$, a collection of text objects, $TO = \{to_1, to_2..., to_m\}$, and also a mapping $\delta : TO \rightarrow 2^C$ that describes the correct classes a text object is supposed to be classified to. *LiveClassifier* aims at finding a one-to-one mapping scheme $\xi : TO \rightarrow 2^C$ such that the size of correct result set $CRS = \{to_i | to_i \in TO, \xi(to_i) = \delta(to_i)\}$ is maximal.

The architecture of the system is illustrated in Figure 1. *LiveClassifier* was designed to interact with both human users and Web applications. The input of *LiveClassifier* includes topic hierarchies and texts that need to be classified, the former for the training phase and the latter for the testing phase. We summarize each component of the system.

- **Feature Extractor**: this component interacts with search engines and extracts highly-ranked search snippets as effective feature sources. It outputs feature vectors to describe both the topic classes and the text objects.

- **HCQF Classifier Generator**: HCQF is the acronym of *Hier-Concept-Query-Formulation*, a technique we developed to train statistical models for topic classes. As its name suggests, it emphasizes on using the concepts embedded in topic hierarchies to train the classifiers. This component interacts with the **Feature Extractor** to organize the class models. It outputs class models to be operated upon by **Text Classifier**.

- **Text Classifier**: Using trained classes output by **HCQF classifier generator**, this component determines proper classes for texts of concern.

2.2 Feature Extractor

To decide the similarity between a text object and a target class, we need a representation model to describe them. In the case of full articles or short documents, we can directly use its content words as its feature source. However, if the text object of concern is a text segment, such as a user query, a named entity, and a topic term and so on, the problem is slightly complicated. In the former case, the text object itself affords abundant feature terms, as it contains tens to thousands of words, while in the later case, the few words composing the text segment are obviously insufficient. To overcome this problem, we send the text segment as a query to search engines and use the returned pages as its feature source. Note also that we use only the snippets as the source, instead of the whole Web pages, so as to reduce the number of page accesses.

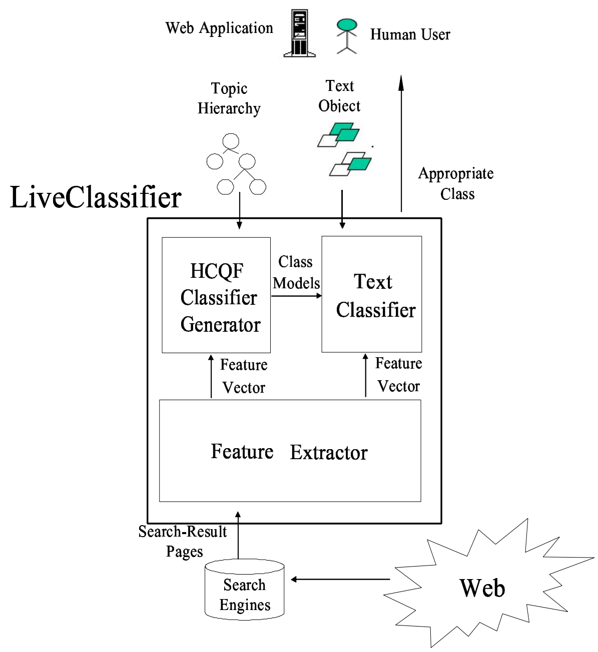

Figure 1: A diagram presenting the architecture of *LiveClassifier*.

When we train the classifiers in Section 2.3, a similar process is repeated: sending the boolean expression of class names to search engines and using the returned pages as the training corpus. Considering the heterogenous nature of the Web, one may doubt whether it is a sound strategy to use the Web resources as the feature source. However, thanks to the recent advances in search technology, we think that, to a certain degree, the highly-ranked returned pages contain quite relevant information and can be treated as an approximate description of the text segment or the topic class. We shall compare the performance of using supervised (hand-labelled) corpus and that of unsupervised corpus composed of search-results in Section 3.

We adopt the vector-space model to describe the features for both text objects and topic classes. Specifically, as we shall present in Section 2.3, **HCQF Classifier Generator** outputs a set of class objects for each separate topic class; both these class objects and text objects are to be converted into vectors to estimate the similarity between them.

To find enough features for text segments and to acquire the training corpora for classes, we formulate queries based on the the text segments or some boolean expression of class names. Suppose that, for each query q, we collect up to N_{max} search-result snippets, denoted as SRS_q. Each SRS_q can then be converted into a bag of feature terms by applying normal text processing techniques, e.g., removing stop words and low-frequency words, to the contents of SRS_q. Let T be the feature term vocabulary, and let t_i be the i-th term in T. With simple processing, a query q can be represented as a term vector v_q in a $|T|$-dimensional space, where $v_{q,i}$ is the weight t_i in v_q. The term weights in this work are determined according to one of the conventional *tf-idf* term weighting schemes [19], in which each term weight $v_{q,i}$ is defined as

$$v_{q,i} = (1 + \log_2 f_{q,i}) \times \log_2(n/n_i),$$

where $f_{q,i}$ is the frequency t_i occurring in v_q's corresponding feature term bag, n is the total number of class objects, and n_i is the number of class objects that contain t_i in their corresponding bags of feature terms. The similarity between a text segment and a class object is computed as the cosine of the angle between the corresponding vectors, i.e.,

$$sim(v_a, v_b) = \cos(v_a, v_b).$$

If, instead of a text segment, the text object is a full article or a short document, its content words are directly treated as SRS_q and the similar processing technique and weighting scheme is operated upon it. We omit the repetitions.

2.3 HCQF Classifier Generator

For the sake of clarity, we present the technique *Hier-Concept-Query-Formulation* (**HCQF**) in a step-by-step manner. A topic hierarchy $TH = (C, R)$ consists of two parts: a set of topic classes $C = \{c_1, c_2..., c_n\}$ and a set of relations $R = \{r_1, r_2..., r_m\}$, relating them hierarchically so that super classes conceptually subsume sub classes. A topic class, whose name is an assigned keyword, essentially represents an abstract concept and the concept is usually embodied by a pre-arranged training set that describes its characteristics. For each c_i, if disregarding its relative position in TH, we can send the name of c_i to search engines and use the returned snippets as its training set. We refer to a concept described by such a training set as *general concept* $G(c_i)$ of c_i ; however, in our case, we think this kind of *general concept* is not the concept that c_i is really meant to represent. The reason is that a *general concept* does not fully reflect the structural information inherent in the topic hierarchy TH.

To remedy this problem, therefore, we define *specific concept* $S(c_i)$ that we think is really the concept c_i represents in the context of a hierarchy. Our idea can be illustrated by an example: suppose Y department is a sub class of X university in some topic hierarchy TH, $G(Y)$ only expresses Y but fails to indicate that Y department is a child class of X university. Instead, the concept that Y really represents in TH is not only about Y but also the fact that Y is a child of X. Put another way, suppose we wish to train the class "CS department," a subclass of "Stanford," most snippets gotten from the query "CS department" positively contain information about "CS department," however, many of them are possibly about "CS department" of some universities other than "Stanford". Such a training set apparently isn't a precise description of the class we wish to train.

Back to the first example, in our research, Y's *specific concept* $S(Y)$ should be the result of Y's *general concept* $G(Y)$ constrained by its parent X. Following this line of reasoning, not only X, but also all of Y's ancestors should exert some influence on Y's *general concept*. Naturally, one is led to think of the converse. Descendant classes should also exert a reciprocal influence on ancestor classes, i.e. descendant classes should enrich the concept of ancestor classes so as to give a more precise description for them. Suppose X university has three departments, Y, Y', and Y'', the concept that X represents is not only about X itself but also the fact it is the parent of Y,Y' and Y''. As above, all descendent classes should enrich the concept of X. Thus, to summarize, for each c_i, the content of a *specific concept* is determined by the combination of three factors: its ancestors, its decedents, and its own *general concept*.

We now formally define what a *specific concept* is. Given some class c_α, whose ancestors are A_{c_α} and whose descendants are D_{c_α}, its *specific concept*, $S(c_\alpha)$, is the union of two separate parts: *specific ancestral concept*, $S_A(c_\alpha)$ and *specific descendant concept*, $S_D(c_\alpha)$, the former being its *general concept* constrained by its ancestors, while the latter being the unification of the *specific ancestral concepts* of all its descendants [1]:

$$S_A(c_\alpha) = G(c_\alpha) \cap \{G(a_i)|a_i \in A_{c_\alpha}\},$$

$$S_D(c_\alpha) = \cup\{S_A(d_j)|d_j \in D_{c_\alpha}\},$$

$$S(c_\alpha) = S_A(c_\alpha) \cup S_D(c_\alpha).$$

The task of preparing the training set to express $S_A(c_\alpha)$ seems difficult, but fortunately real-world search engines relieve us much of the trouble. One may send the query as a boolean expression c_α and the name of its ancestors A_{c_α} to search engines and use the returned snippets as the required training set for $S_A(c_\alpha)$. Conversely, when preparing the training set to express $S_D(c_\alpha)$, one simply adds up all the training sets for *specific ancestral concept* of D_{c_α}.

In more practical terms, the total training set of class c_α is then composed of the training set (N_{max} snippets) for $S_A(c_\alpha)$ and the training sets ($N_{max} * |D_{c_\alpha}|$) for $S_D(c_\alpha)$. Note that each N_{max} snippets can be then converted to a class object according to the vector space model discussed in Section 3.1. Therefore, c_α is then presented as an array of class objects, one of them is from $S_A(c_\alpha)$ while others are from $S_D(c_\alpha)$.

The strength of **HCQF** lies exactly in this kind of rich training set. For c_α, its concept is not only specified by its ancestors and itself, but also by all its descendants. Suppose we only consider $S_A(c_\alpha)$ but drop $S_D(c_\alpha)$, we have merely N_{max} snippets to train c_i; on the other hand, if we take $S_D(c_\alpha)$ into consideration, we may have a dramatic $(1 + |D_{c_\alpha}|) * N_{max}$ snippets to train c_α. (Note that since we convert the training sets into class objects, the above expression can be better restated as the comparison between 1 and $1 + |D_{c_\alpha}|$ objects.) Moreover, the additional $|D_{c_\alpha}| * N_{max}$ snippets (or $|D_{c_\alpha}|$ objects) don't contain as much noise as one might expect, because they are already constrained by c_α along with its ancestors beforehand.

It can easily be seen that one may train all the classes in TH by traversing it by the manner of BFS twice. For each c_i, the first round collects the training set for $S_A(c_i)$, while the second round adds up the training sets for $S_D(c_i)$ and $S(c_i)$. Having collected all the necessary training sets, by using the collection of the terms in all training sets as the total feature vocabulary T, these training sets can be converted into class objects.

So far, we have presented the overall idea of **HCQF** and it can be observed that it mainly concerns about how to formulate precise queries and organize the corpus. We now are left with two more problems. First, the leaf class c_{leaf}, unlike internal classes, cannot be strengthened by its descendent classes. In other words, $S_d(c_{leaf}) = \emptyset$ and **HCQF** seems have a weaker descriptive power for them. Second, suppose we are given only a non-hierarchical tree, i.e. a flat structure, can **HCQF** be generalized so as to be applied to it too?

The answer to two above questions lies in the fact that one can always find some classes to enrich a leaf class by inserting them as "pseudo" children classes. Given some leaf class c_{leaf}, when collecting the snippets to organize $S_a(c_{leaf})$, one can easily find some associated terms of c_{leaf} and use some filtering mechanisms to choose proper terms as child classes of c_{leaf} [2].

[1]We use the set operations to express our idea, and their meaning will be made clear in the following discussion. Strictly speaking, the notation we use here is not mathematically rigorous. The "concept," actually, isn't composed of distinct entities as a mathematical set is.

[2]It is not necessary that one always finds the associated terms from the Web-retrieved snippets. If the leaf class has some local training document set, one can also extract associated terms from it.

HCQF$(TH = (C, R))$
C: the set of topic classes
R: relations among topic classes
SRS: the collection of training sets (search result snippets) (initially ø)
CO: the collection of feature vector of of class objects

1: **for all** $c \in C$, according to R, choose c by BFS **do**
2: $SRS_c \leftarrow$ send $c \cup c_{ancestors}$ to search engines
3: **if** c= leaf **then**
4: $AT_c \leftarrow$ Associated_Terms_by_Subsumption(c, SRS_c)
5: **for all** $at \in AT_c$ **do**
6: $SRS_c \leftarrow SRS_c \cup$ send $(c \cup at)$ to search engines
7: **for all** $c \in C$, according to R, choose c by BFS **do**
8: $SRS_c \leftarrow SRS_c \cup SRS_{c_{descendents}}$
9: $SRS \leftarrow SRS \cup SRS_c$
10: **for all** $srs \in SRS$ **do** {transform SRS into a feature vector according to Section 3.1}
11: $co \leftarrow$ transform srs
12: $CO \leftarrow CO \cup co$
13: return CO

Associated_Terms_by_Subsumption(c, SRS_c)
c: topic class
SRS_c: training set of c
AT: the set of associated terms (initially ø)

14: **for all** $t \in SRS_c$ **do**
15: **if** $p(c|t) \geq 0.8$ and $p(t|c) < 1$ **then**
16: $AT \leftarrow AT \cup t$
17: return AT

Associated_Terms_by_Co-occurrence(c, SRS_c)
c: topic class
SRS_c: training set of c
AT: the set of associated terms (initially ø)

18: **for all** $t \in SRS_c$ **do**
19: **if** $DF(t, c)/DF(C) > \varepsilon$ **then** {DF value can be gotten from search engines}
20: $AT \leftarrow AT \cup t$
21: return AT

Figure 2: An algorithmic procedure describing HCQF. Note that Line 4 can be replaced by Associate_Term_by_Co-occurrence(c, SRS_c).

In our experiments, we have employed the following two techniques to create the "pseudo" child classes. We choose either (1) the terms subsumed by c_{leaf} [20]; or (2) the terms having the highest mutual information with c_{leaf}.

The first technique is based on the assumption that, suppose a term c^d_{leaf} is subsumed by c_{leaf}, the documents that c^d_{leaf} appears always (or almost always) contain c_{leaf}, while the documents containing c_{leaf} do not necessarily contain c^d_{leaf}. The formula is set as follows:

$$P(c_{leaf}|c^d_{leaf}) \geq \theta,$$

$$P(c^d_{leaf}|c_{leaf}) < 1.$$

[20] set θ to 0.8. However, in our experience, a slightly better result can be acquired by setting θ to 0.85 for Web documents.

The second technique is based on the assumption that the concept of c_{leaf} can be enriched by its most relevant terms too. We choose the terms that appear with c_{leaf} above a certain threshold of times. We denote the document frequency of some term t as $DF(t)$ and that of the co-occurrence of s and t as $DF(t, s)$, then our idea can be expressed as

$$\frac{DF(t, c_{leaf})}{DF(c_{leaf})} > \epsilon$$

Based on heuristics, we set ϵ to 0.45 in this work. The whole algorithmic procedure of **HCQF** is presented in Figure 2.

TextClasssifier(to, C, CO, n)
to: the unknown text object
C: the set of classes
CO: the set of class objects returned by **HCQF**
n: the number of target categories

for all $co \in CO$ **do**
$r(to, co) \leftarrow sim(v_{to}, v_{co})$
$R_k(to) \leftarrow$ the k class objects $co \in CO$ with highest $r(to, co)$ scores
for all $c \in C$ **do**
$r_{kNN}(to, c) \leftarrow 0$
for all $co \in R_k(to)$ **do**
for all c is a class of co **do** {a class object co may enrich multiple ancestor classes}
$r_{kNN}(to, c) \leftarrow r_{kNN}(to, c) + r(to, co)$
return top-ranked n classes in C according to the decreasing order of $r_{kNN}(to, c)$

Figure 3: An algorithmic procedure describing the text classification process.

2.4 Text Classifier

Given a new candidate text object to, **Text classifier** determines a set of classes that are considered as to's most relevant classes.

As discussed in Section 2.2, the candidate text object to is represented as a feature vector v_{to}. For the classification task, we here adopt a kNN approach.

kNN has been an effective classification approach to a broad range of pattern recognition and text classification problems [9]. By kNN approach, a relevance score between text object to and candidate class C_i is determined by the following formula:

$$r_{kNN}(to, C_i) = \sum_{v_j \in R_k(to) \cap C_i} sim(v_{to}, v_j)$$

where $R_k(to)$ are to's k most-similar class objects, measured by sim function, which is the cosine angle between the two vectors, in the whole collection. Figure 3 shows the algorithmic procedure of this classification process.

The classes that a text object is to be assigned to are determined by either a predefined number of most-relevant clusters or a threshold to pick those clusters with scores higher than the specified threshold value. Different threshold strategies have both advantages and disadvantages [26]. In this study, to evaluate the performance, we select the five most relevant classes as candidates.

3. EXPERIMENTS

Having described the overall idea of *LiveClassifier*, we now try to justify it by empirical evidence. In designing the experiments, we not only assessed the accuracy of *LiveClassifie*, but also explored possible applications that could be derived from it.

Throughout the experiments, we use Yahoo!'s topic hierarchy as the testing bed. k is set to 5. The search engine employed was Google.

To better evaluate how *LiveClassifier* performs when the length of the test text object varies, we divided them into three groups: *text segments, short documents* and *full articles*. Text segments were directory names of lower levels classes. For example, in the following Computer Science experiment, the directory names "Intelligent Software Agent" and "Fuzzy Logic" could be taken as text segments of concern and were supposed to be classified into the higher level class "Artificial Intelligence". For each directory in Yahoo!, there was a list of Web sites accompanied by site description offered by Yahoo!'s indexers. We used the Web pages of the

Table 1: Top 1-5 inclusion rates of classifying text objects into second level classes of CS-tree from Yahoo! under various circumstances.

Topic Hierarchy Used + Corpora Source	Approach	Text Type	Top-1	2	3	4	5
Manual Hierarchy + Unsupervised Web Corpora	Based on only second level classes (Approach 1)	Full Article	.3389	.3785	.6045	.6214	.6779
		Short Document	.5780	.7008	.8034	.8146	.8483
		Text Segment	.4917	.6346	.6943	.7242	.7545
	Based on first three level classes (Approach 2)	Full Article	.5367	.6780	.7288	.7458	.7514
		Short Document	.7837	.8932	. 9326	.9326	.9606
		Text Segment	.7384	.8775	.9272	.9371	.9636
Augmented Hierarchy + Unsupervised Web Corpora	Based on pseudo classes generated by subsumption technique plus classes at the first two levels (Approach 3)	Full Article	.3785	.5254	.6384	.7005	.7231
		Short Document	.6753	.8432	.8432	.8932	.8932
		Text Segment	.6060	.7252	.8146	.8411	.8510
	Based on pseudo classes generated by co-occurrence technique plus classes at first two levels (Approach 4)	Full Article	.4067	.6045	.7062	.7457	.7740
		Short Document	.7050	.8034	.8764	.8932	.8932
		Text Segment	.6424	.7417	.8245	.8510	.8609
Not Using Topic Hierarchy + Supervised Corpora	Using the short documents of Yahoo! (Approach 5)	Full Article	.3785	.5706	.6497	.7006	.7288
		Short Document	.6142	.6751	.7005	.7100	.7259
		Text Segment	.6912	.8039	.8671	.9003	.9202

sites as full articles [3] and the description of the sites as short documents.

3.1 The Overall Performance of LiveClassifier

We first focused on how well *LiveClassifier* performs when dealing with text objects of different lengths. We chose a specific domain, computer science in Yahoo! Computer Science topic hierarchy to conduct this experiment. There were totally 36 second-level, 177 third-level, and 278 fourth-level classes, all rooted at the class "Computer Science". We used the second-level classes, e.g., "Artificial Intelligence" and "Linguistics" as the target classes and tried to classify text objects into them.

For text segments, the 278 fourth-level classes were used as test instances. Also, we chose randomly 177 full articles and their corresponding short documents from the fourth-level classes.

In general, we were interested in the following questions and designate them respectively as Approaches 1-5:

1. Suppose we use only the restricted version of **HCQF**, i.e. dropping $S_D(C)$ and only using $S_A(C)$, remove the root and the third-level classes, and don't consider generating pseudo classes, how well does **HCQF** do? This is equivalent to using only a flat structure and can be thought of as the bottom-line. (Approach 1)

2. Suppose we take both $S_D(C)$ and $S_A(C)$ into consideration but still don't generate pseudo classes automatically, how well does **HCQF** do? (Approach 2)

3. Suppose we are not given third-level classes, how well does **HCQF** do if it generates pseudo classes automatically by the subsumption technique? (Approach 3)

4. The same situation as (3), but now **HCQF** generates pseudo classes by the co-occurrence technique. (Approach 4)

5. Instead of using **HCQF**, the short documents of Level 2 and Level 3 classes are used as training corpora, how well do the classifiers perform? (Approach 5)

[3]Note that we treated the Web pages in their simplest form, i.e. only their full content without considering any tag information.

Note that in Approaches 3 and 4, for each target class, we deliberately made the number of its pseudo child classes the same with Approach 2 so as to evaluate the performance of **HCQF** in the context of a manually-constructed hierarchy and an automatically-augmented hierarchy.

Table 1 shows the result of the achieved top 1-5 inclusion rates, the top n inclusion rate is the rate of test objects whose highly ranked n candidate classes contain the correct class. From this table, it can be observed that Approach 2 surpassed all other approaches. This is a hint that a well-organized topic hierarchy greatly contributes to the high performance of **HCQF**. Approaches 3 and 4 also got promising results, though not as good as Approach 2. This indicates that both subsumption technique and co-occurrence technique could get proper pseudo classes; **HCQF** does not necessarily rely on user-defined topic hierarchies; to a certain degree, the manual-defined topic hierarchies can be approximated by **HCQF**'s automatic mechanisms. The worst was Approach 1, a simple flat structure. However, it deserves attention that, though it fell far behind other approaches, considering that there were totally 36 classes, its overall performance was still acceptable. This not only revealed the superiority of Web resources but also implicitly suggested that, to train a classifier, a simple but effective method is to simply designate a set of distinct class names.

A very interesting observation can be made about Approach 5. Using Yahoo!'s short documents as the training corpora also got decent results, second only to Approach 2 but superior to Approaches 3 and 4. Unlike the previous four Approaches using unsupervised training corpora downloaded from Search engines, Approach 5 can be deemed as using supervised hand-labelled training corpora. If unsupervised training corpora could get comparable (Approaches 3 and 4) or even better result (Approach 2) than supervised training corpora, it implies that a topic-hierarchy composed of keywords specified by indexers or librarians seems enough to create the needed classifiers, rather than manually labelling a lot of corpus.

Concerning the text type axis, it can be observed that the three types all got satisfactory results. In general, the classifier we trained could categorize text segments and short documents with promising accuracy. This confirmed our conjecture that Web search-result snippets were a proper description of the text segments and could be used as the feature source.

Compared to short documents and text segments, the full article experiment didn't get as good results, though the results were still promising. The probable reason of this performance degradation was that the content of the test Web pages was too diverse so that they sometimes were not conceptually closely related to the target class.

3.2 Granularity and Diversity

Having observed how **HCQF** performed in a specific area: Computer Science, we then tried to examine whether **HCQF** could be applied to a topic-hierarchy of more diverse domains and of deeper depth. We extracted parts of of Yahoo!'s directory about "Science" and "Social Science" of 5 level deep. There were totally 84 text segments and 139 short documents and their corresponding full articles in Level 5.

Table 2: The information of Science and Social Science hierarchies extracted from Yahoo!'s directory.

Level 2	Level 3	Level 4
Science	Mathematics	Geometry, Number Theory
	Chemistry	Chemist, Chemical and Biological Weapons
	Astronomy	Solar System, Cosmology
Social Science	History	Historiology, Genealogy
	Sociology	Social Class and Stratification, Urban Studies
	Linguistics and Human Languages	Translation and Interpretation, Word and Wordplay

Unlike the preceding experiment, we tried to classify the various text objects of Level 5 into Levels 2, 3, and 4 respectively. Classifying text objects into different levels of the topic hierarchy has a consequential implication: it means whether HCQF can create thematic information of different degrees of refinement. In particular, the depth of a class in a topic hierarchy suggests its own topicality and speciality. And if a text object can be successfully classified into classes of different levels, it means much more information can be thus created.

Table 3: The Top 1 inclusion rate of classifying text objects into different levels. Number of pseudo classes created by Approaches 3 and 4 is 6.

Approaches	Text Types	Level 2	Level 3	Level 4
Approach 1	Full Article	.5731	.4219	.6094
	Short Document	.4748	.3525	.5683
	Text Segment	.4609	.5714	.8810
Approach 2	Full Article	.8047	.6094	N/A
	Short Document	.8417	.6978	N/A
	Text Segment	.9634	.8333	N/A
Approach 3	Full Article	.6172	.4667	.6171
	Short Document	.5467	.4317	.6384
	Text Segment	.6190	.6428	.9166
Approach 4	Full Article	.5390	.6453	.6610
	Short Document	.5755	.4101	.6834
	Text Segment	.5827	.6310	.9048
Approach 5	Full Article	.6562	.5313	.8203
	Short Document	.5683	.4453	.9496
	Text Segment	.8452	.7142	.4048

Table 2 lists the details about the Level 2 to Level 4; Table 3 lists the results. It can be observed that classifying text objects into different levels of the topic hierarchy got roughly the same results. Although the higher the levels, the number of classes was smaller and it seems easier to classify text objects into them, this factor was cancelled by another fact: the concept of higher level classes were harder to be trained correctly due to their generality and abstraction.

3.3 Creating Thematic Metadata for Textual Data

Recent advances in text processing technologies, such as text pattern recognition, information extraction, metadata annotation can extract metadata (facts) about people, place, time from texts with high accuracy. However, the metadata created by these kinds of technologies, is still too primitive to be used as a basis for more advanced applications, such as concept-based search. How to create more refined metadata with limited human intervention is a problem that deserves investigation. In this experiment we explored the possibility of using *LiveClassifier* to help create more refined metadata.

We extracted three hierarchies from Yahoo!, respectively "People" (People/Scientist), "Place" (Region/Europe), and "Time" (History-time Period). For these three cases, we randomly selected 100, 100, and 93 class names, which could be considered as a kind of text segment, from the bottom-level and assigned them onto the second-level classes. And as before, we randomly selected 77, 80, 45 short documents along with the corresponding full articles from Yahoo! to conduct the experiments.

Table 5 lists some samples of the test text segments and their corresponding classes. Table 4 lists the classification results for various types of text. It could be observed that in the "People" and "Place" cases, our approach got very satisfactory results, while in the "Time" case we did not get similar good results. The reason for its performance degradation seems that the concept of a time period, such as "Renaissance"and "Middle Ages", is too broad and too much noise is contained in the returned snippets, thus lowering the precision of classification.

On the contrary, the high performance of the "People" case and "Place" case is contributed by two factors: (1) The concepts of the classes themselves are very specific. A specific concept implies that Web search results are very precise and coherent and thus have a higher chance of training the class correctly. (2) The classes are themselves very distinct from one another. Notice especially this factor can partly explain why the "People" and the "Place" cases got better results than the above CS experiment. The fields of CS often overlap in subjects while people's jobs and the places seldom do.

Table 5: Some samples of the test text segments and their corresponding classes extracted from Yahoo!.

	Samples and Corresponding Second-level Classes	
People	Curie, Marie (1867-1934)	Physicists
	Korzybski, Alfred (1879-1950)	Linguists
	Fulton, Robert (1765-1815)	Engineers&Inventors
	Cantor, Georg (1845-1918)	Mathematicians
Place	Piraeus	Greece
	Kannus	Finland
	Vorchdorf	Austria
	Grindavik	Iceland
Time	Glorious Revolution	17th Century
	Peloponnesian War	Ancient History
	Hanseatic League	Middle Ages
	French Revolution	18th Century

Table 4: Top 1-5 inclusion rates for classifying Yahoo!'s People, Place, and Time text objects.

	Approach	Text Type	Top-1	2	3	4	5
Yahoo! (People)	Approach 1	Full Article	.5866	.6933	.7466	.8	.8
		Short Document	.8961	.8961	.8961	.8961	.8961
		Text Segment	.8654	9808	.9808	.9904	.9904
	Approach 2	Full Article	.7066	.8133	.8533	.8533	.8533
		Short Document	.8533	.8961	.8961	.8961	.8961
		Text Segment	.8846	.9808	.9904	.9904	.9904
Yahoo! (Place)	Approach 1	Full Article	.7375	.8375	.85	.875	.8875
		Short Document	.9000	.9250	.9500	. 9750	.9750
		Text Segment	.8700	.9500	.9700	.9700	.9800
	Approach 2	Full Article	.8625	.8875	.8875	.9	.9125
		Short Document	.9500	.9750	.9750	.9750	.9750
		Text Segment	.9200	.9600	.9700	.9700	.9800
Yahoo! (Time)	Approach 1	Full Article	.3333	.4444	.5555	.6444	.6444
		Short Document	.4	.5555	.6	.6444	.7333
		Text Segment	.1612	.3225	.4301	.4838	5591
	Approach 2	Full Article	.4222	.4444	.5555	.6444	.6444
		Short Document	.4444	.5555	.6222	.6666	.7555
		Text Segment	.3854	.5521	.6354	.6562	.6562

Table 6: The information of the paper data set.

Conference	# Papers	Assigned Class
AAAI'02	29	CS:Artificial Intelligence
ACL'02	65	CS:Linguistics
JCDL'02	69	CS:Lib. & Info. Sci.
SIGCOMM'02	25	CS:Networks

Table 7: Top 1-5 inclusion rates for classifying paper titles.

Approach	Top-1	2	3	4	5
Approach 1	.2021	.2872	.3457	.3777	.4255
Approach 2	.4628	.6277	.7181	.7713	.8085

3.4 Paper Title Classification

We mentioned in Section 1 that one could design a Web information service that collects academic papers and use a classification technique to determine the specialized fields of researchers. We now try to use *LiveClassifier* to show that this goal is achievable.

In this experiment, we collected a data set of academic paper titles from four computer science conferences in year 2002 and tried to classify them into the 36 second-level CS classes again. Each conference was assigned to the Yahoo! class to which the conference was considered to belong, e.g., AAAI'02 was assigned to "Computer Science/Artificial Intelligence," and all the papers from that conference unconditionally belonged to that category. Table 6 lists the relevant information of this paper data set. Note that this might not be an absolutely correct classification strategy, as some papers in a conference may be even more related to other domains than the one we assigned them. However, to simplify our experiment, we made this straightforward assumption. Table 7 lists the experimental results. Also, Table 8 displays some examples of miss-classified papers. It can be observed that the contents of these miss-classified papers were actually more related to the classes assigned.

4. RELATED WORK

The fundamental similarity between **HCQF** and automatic query expansion techniques is not hard to be discerned. The latter technique has been studied for decades with debatable degrees of success; for a summary article, see [23]; more recent developments can be found in [25, 18, 3]. Query expansion was first introduced to overcome the problem of word mismatch, a problem fundamental to Information Retrieval. In a manner of speaking, the topic hierarchy defined by users can be taken as a kind of thesaurus; but the topic hierarchy represents the subsumption relationship among the concept of the classes rather than some semantical relationship.

A lot of Web IE systems have been developed and met different degrees of success. The following list we cite is bound to be incomplete [10, 12, 11, 16, 4, 13]. However, to the best of our knowledge, the possibility of combining text classification technique with Web IE techniques to create more advanced Web information services seems seldom to get a direct treatment in the literature.

Using the Web as a super huge knowledge source to solve problems is common practice these days. There have been attempts of using Web Mining techniques to extract templates [2], to disambiguate word sense [1], to resolve PP attachment [24], to translate terms [14], to build query taxonomies [5], and to categorize documents [8].

The works most closely related to ours are [17, 15]. Both works were devised in view of overcoming the problem of expensiveness and scarcity of hand-labelled corpora, although their approach and ours are quite different in methodological aspect. Their main idea is to use a bootstrapping process to label the unlabelled documents probabilistically, and use the newly-labelled corpus to help retrain the classifier and recursively so. The assumption of both works is that an initial data set already exists, which may be some labelled corpus [17], or some manually-assigned keywords [15]. They focus on the training stage, i.e., how to optimize the classifier based on known corpus in the training stage, while in this work we focus on how to prepare a more suitable and rich initial data set. We think their works and ours are complementary and it is possible to upgrade the performance of *LiveClassifier* by adopting their technique. Also, more refined query expansion techniques can be incorporated into **HCQF** to creating more suitable pseudo classes.

5. CONCLUDING REMARKS

In this work, we have presented a system that can automatically extract corpora from the Web to train classifiers. The main merits of *LiveClassifier* are its wide adaptability and its flexibility. The needed classifier can be created by defining a topic hierarchy. The necessary corpora can be fetched and organized automatically, promptly, and effectively. Furthermore, the performance of the classifiers created are in general good, supported by empirical evidence.

From the perspective of application, *LiveClassifier* can create more information at thematic level and this information can in turn

Table 8: Selected examples of miss-classified paper titles.

Paper Title	Conference	Target Class	Top-1	2	3	4	5
A New Algorithm for Optimal Bin Packing	AAAI	AI	ALG	AI	MOD	COLT	DNA
(Im)possibility of Safe Exchange Mechanism Design	AAAI	AI	NET	SC	LG	DB	MD
Performance Issues and Error Analysis in an Open-Domain Question Answering System	ACL	LG	AI	LG	ALG	DC	SC
Active Learning for Statistical Natural Language Parsing	ACL	LG	AI	LG	NN	COLT	ALG
Improving Machine Learning Approaches to Coreference Resolution	ACL	LG	AI	LG	ALG	FM	NN
A language modelling approach to relevance profiling for document browsing	JCDL	LIS	AI	UI	LG	LIS	ALG
Structuring keyword-based queries for web databases	JCDL	LIS	AI	LIS	DB	ALG	ARC
A multilingual, multimodal digital video library system	JCDL	LIS	LG	UI	LIS	ECAD	NET
SOS: Secure Overlay Services	SIGCOMM	NET	SC	NET	MC	OS	DC

AI :Artificial Intelligence
ALG :Algorithms
ARC :Architecture
COLT:Computational Learning Theory
DB :Databases
DC :Distributed Computing
DNA :DNA-Based Computing
ECAD:Electronic Computer Aided Design
FM :Formal Methods
LG :Linguistics
LIS :Library and Information Science
MC :Mobile Computing
MOD:Modeling
NET :Networks
NN :Neural Network
OS :Operating Systems
SC :Security
UI :User Interface

Table 9: Yahoo!'s Computer Science experiment when the corpus size increases. Approach 1.

N_{max}	Text Type	Top-1	2	3	4	5
100	Full Article	.3389	.3785	.6045	.6214	.6779
	Short Document	.5780	.7008	.8034	.8146	.8483
	Text Segment	.4917	.6346	.6943	.7242	.7545
200	Full Article	.5311	.6271	.6723	.6949	.7118
	Short Document	.5780	.6678	.7008	.7409	.8034
	Text Segment	.4850	.6213	.6910	.7243	.7409
400	Full Article	.4294	.5028	.5593	.6102	.6251
	Short Document	.5563	.6632	.6803	.7423	.8011
	Text Segment	.4518	.5880	.6545	.6910	.7043
600	Full Article	.4294	.5198	.5593	.5819	.5875
	Short Document	.5454	.6553	.6731	.7004	.7321
	Text Segment	.4219	.5747	.6445	.6678	.6810
800	Full Article	.4294	.5198	.5593	.5819	.5875
	Short Document	.5450	.6345	.6855	.6921	.6999
	Text Segment	.4219	.5083	.5648	.6047	.6146

be used to create more value-added Web information services. For common human users, *LiveClassifier* also bestows much convenience. No longer troubled by the tedious work of preparing corpora, users may effortlessly construct many classifiers by his/her own preference.

The effectiveness of *LiveClassifier* deserves some remarks. As discussed in the preceding section, downloading un-labelled Web corpora to augment features or to enhance the size of training corpora has been tried in many recent works. However, few have considered the problem of "how" to collect and organize the corpora.

One may entertain the idea that **HCQF** simply depends on the enormous size of Web resource to train the topic-hierarchy, however, this is not the case. Table 9 lists the results of the Computer Science experiment when training corpora increased. It can be observed that the performance *did not* ameliorate with the size of the training corpora, on the contrary, it is the other way around.

A probable reason of this phenomenon is that the lowly-ranked snippets contain much more noise, thus dragging down the performance. Obviously, downloading Web documents indiscriminately does not ensure success in training. The reason that **HCQF** can get better results is rather its exploiting structural information contained in topic hierarchies. We have presented in Section 3 that subtrees of limited depth extracted from Yahoo!'s directory can achieve satisfactory results. We have also proven in Section 3.2 that in different granularities and in diverse domains, **HCQF** can achieve acceptable results. However, designing experiments of larger scale is still desirable.

LiveClassifier can be accessed online in the following URL http://liveclassifier.iis.sinica.edu.tw/. Users can create and modify classifiers online.

6. REFERENCES

[1] E. Agirre, O. Ansa, E. Hovy, and D. Martinez. Enriching very large ontologies using the www. In *Proceedings of ECAI 2000 Workshop on Ontology Learning*, 2000.

[2] Z. Bar-Yossef and S. Rajagopalan. Template detection via data mining and its applications. In *Proceedings of the 11st International World Wide Web Conference*, pages 26–33, 2002.

[3] C. Carpineto, R. De Mori, G. Romano, and B. Bigi. An information-theoretic approach to automatic query expansion. *ACM Transactions on Information Systems*, 19(1):1–27, 2001.

[4] C.-H. Chang and S.-L. Lui. Iepad: information extraction based on pattern discovery. In *Proceedings of 10th International World Wide Web Conference*, pages 681–688, 2001.

[5] S.-L. Chuang and L.-F. Chien. Towards automatic generation of query taxonomy: A hierarchical query clustering approach. In *Proceedings of the 2nd IEEE International Conference on Data Mining*, pages 75–82, 2002.

[6] W. Cohen and W. Fan. Learning page-independent heuristics for extracting data from web pages. In *Proceedings of the 8th International World Wide Web Conference*, 1999.

[7] W. Cohen, M. Hurst, and L. Jensen. A flexible learning system for wrapping tables and lists in html documents. In *Proceedings of the 11th International World Wide Web Conference*, pages 232–241, 2002.

[8] W. Cohen and Y. Singer. Context-sensitive learning methods for text categorization. In H.-P. Frei, D. Harman, P. Schäuble, and R. Wilkinson, editors, *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 307–315, Zürich, CH, 1996. ACM Press, New York, US.

[9] B. V. Dasarathy. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. McGraw-Hill Computer Science. IEEE Computer Society Press, Las Alamitos, California, 1991.

[10] E. O. Doorenbos, R. and D. Weld. A scalable comparison-shopping agent for the world-wide web. In *Proceedings of Autonomous Agents*, pages 39–48, 1997.

[11] C.-N. Hsu and M.-T. Dung. Generating finite-state transducers for semi-structured data extraction from the web.

Journal of Information Systems, Special Issue on Semistructured Data, 23(8):521–538, 1998.

[12] N. Kushmerick, D. Weld, and R. Doorenbos. Wrapper induction for information extraction. In *International. Joint Conference on Artificial Intelligence*, pages 729–737, 1997.

[13] B. Liu, R. Grossman, and Y. Zhai. Mining data records in web pages. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.

[14] W.-H. Lu, L.-F. Chien, and H.-J. Lee. Anchor text mining for translation of web queries. In *Proceedings of the first IEEE International Conference on Data Mining*, pages 401–408, 2001.

[15] A. McCallum and K. Nigam. Text classification by bootstrapping with keywords. In *ACL Workshop for Unsupervised Learning in Natural Language Processing*, 1999.

[16] I. Muslea, S. Minton, and C. Knoblock. Stalker: Learning extraction rules for semistrctured, web-based information sources. In *Workshop on AI and Information Integration, in conjunction with the 15th National Conference on Artificial Intelligence (AAAI-98)*, 1998.

[17] K. Nigam, A. K. McCallum, S. Thrun, and T. M. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.

[18] Y. Qui and H. Frei. Concept based query expansion. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 160–169, 1993.

[19] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24:513–523, 1988.

[20] M. Sanderson and W. B. Croft. Deriving concept hierarchies from text. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 206–213, 1999.

[21] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1), 2002.

[22] S. Soderland. Learning to extract text-based information from the world wide web. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, pages 251–254, 1997.

[23] K. Sparck-Jones. Notes and references on early classification work. *SIGIR Forum*, 25(1):10–17, 1991.

[24] M. Volk. Exploiting the www as a corpus to resolve pp attachment ambiguities. In *Proceedings of Corpus Linguistics*, 2001.

[25] J. Xu and W. Croft. Query expansion using local and global document analysis. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 412–420, 1996.

[26] Y. Yang. A study on thresholding strategies for text categorization. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 137–145, 2001.