Capturing Evolving Patterns for Ontology-based Web Mining

Yuefeng Li and *Ning Zhong
School of Software Engineering and Data Communications, Queensland University of Technology, Australia
[*]Department of Systems and Information Engineering, Maebashi Institute of Technology, Japan
E-mail: y2.li@qut.edu.au, [*]zhong@maebashi-it.ac.jp

Abstract

An ontology-based Web mining model tends to extract an ontology from user feedback and use it to search the right data from the Web to answer what users want. It is indubitable that we can obtain numerous discovered patterns using a Web mining model. However, some discovered patterns might include uncertainties when we extract them. Also user profiles are changeable. Therefore, the difficult issue is how to use and maintain the discovered patterns. This paper presents a theoretical framework for this issue, which consists of automatic ontology extraction, reasoning on the ontology and capturing evolving patterns. The experimental results show that all objectives we expect for the theoretical framework are achievable.

Keywords: *Ontology-based Web mining, data reasoning, data mining, ontology learning, pattern evolution.*

1. Introduction

There are two fundamental issues regarding the effectiveness of using the Web data: *mismatch* and *overload*. The mismatch means some interesting and useful data has not been found (or missed out), whereas, the overload means some gathered data is not what users expect.

We argue that the following three steps are necessary to overcome the fundamental issues within Web Intelligence (WI). The first step is to create an adequate environment (e.g. Semantic Web) for users to manipulate their data (e.g., XML). The obvious benefit of such arrangement is that we can decrease the burden of search engines. The second step is the discovery of interesting and useful knowledge from the environment. The third step is data reasoning of discovered knowledge in order to answer what users really want.

Currently, the application of data mining techniques to Web data, called Web mining, is used to discover patterns (knowledge) from Web data [4] [27]. It is indubitable that we can obtain numerous discovered patterns using a Web mining model. However, there still exits a gap between Web mining and the effectiveness of using the Web data, since it is very difficult to use the numerous discovered patterns for dealing with the fundamental issues. One reason is that some discovered patterns might include uncertainties when we extract them. Another reason is that user profiles are changeable.

The problem we especially concern in this paper is data reasoning of discovered patterns in order to eliminate the uncertainties and capturing evolving patterns according to user feedback. Data reasoning of discovered patterns means the use and maintenance of discovered patterns to answer what users really want. The difficult problems in data reasoning are representations of discovered patterns and capturing evolving patterns.

An ontology-based Web mining model tends to overcome the above difficult problems, which uses ontologies to represent the discovered patterns in order to search the right data for what users want [16] [17]. In this paper we present a theoretical framework to extend the ontology-based Web mining model. It consists of automatic ontology extraction, reasoning on the ontology and capturing evolving patterns. The innovation is that the theoretical framework deals with pattern evolution.

Using this theoretical framework, the system can update its ontology by adding new positive patterns, removing some inadequate patterns or even updating some existing positive patterns if they cause an incorrect decision. We can also prove that such evolution can enable the system to incrementally filter out most non-relevant data. Moreover, this theory can decrease the burden of on-line training since it only needs part of the training data.

The remainder of the paper is structured as follows. We begin by introducing some basic definitions about automatic ontology extraction in Section 2. Syntactically we assume that an ontology can be constructed from some primary ones. We also present a new procedure for automatic ontology extraction. In Section 3, we discuss reasoning on the ontology. We also present decision rules and a novel filtering algorithm to answer what users want. In Section 4, we present an algorithm for capturing evolving patterns. In Section 5, we show our experiments for the proposed algorithms. Finally, Section 6 discusses related work and gives conclusions.

2. Automatic Ontology Extraction

Manual ontology engineering is a tedious and cumbersome task that can easily result in a bottleneck for knowledge acquisition [37] [24] [17]. The key issue for this problem is automatic ontology extraction and maintenance (or called ontology learning [23]).

The core structure of an ontology in ontology leaning can be defined as a 5-tuple [23]: $O ::= \langle C, \mathcal{R}, \mathcal{H}^C, rel, \mathcal{A}^O \rangle$, where C is a set of classes (concepts); $\mathcal{R}$ is a set of relations; $\mathcal{H}^C \subseteq C \times C$ is called taxonomy, $\mathcal{H}^C$ (C_1, C_2) means C_1 ì is-aî C_2; $rel : \mathcal{R} \rightarrow C \times C$ is a function defined for other relations; and $\mathcal{A}^O$ is a logical language.

For the existing ontology learning algorithms proposed so far, the backbone of the ontology $\mathcal{H}^C$ only includes ì is-aî relation. We can find more details for modelling the taxonomic relation in [34]. However, the taxonomy is insufficient to construct a complete concept space for user profiles [17]. In this research we extend the backbone to include both ì is-aî relation and ì part-ofî relation.

Syntactically we assume the user concepts (classes) can be constructed from some primary ones. According to this assumption, an ontology consists of *primitive classes*, and *compound classes*, which can be set up from a set of primitive classes using some ì constructorî operations [17]. The primitive classes are the smallest concepts that cannot be assembled from other classes, however they may be inherited by some derived concepts or their children (sub-terms).

Figure 1 shows such a backbone of the ontology, which organises a set of discovered patterns. The set of primitive objects Θ = {*shop*, *pet*, *city*, *accommodation*}. Because *dog* and *cat* are *pets* (also *hotel* is-an *accommodation*}, we use ì is-aî links to show the relation between them and the corresponding primitive objects.

Assume there are 3 regular patterns discovered from relevant documents in the training set, which are represented as sets of keyword-frequency pairs:

d_1 = {(*dog*, 4), (*shop*, 6)},
d_2 = {(*cat*, 5), (*shop*, 15)},
d_3 = {(*pet*, 3), (*shop*, 7), (*city*, 10)}.

Using the inheritance (or is-a relation), we can obtain two compound objects: p_1 and p_2 (see Figure 1) from d_1, d_2 and d_3, where $d_1 \rightarrow p_1$, $d_2 \rightarrow p_1$ and $d_3 \rightarrow p_2$. We call p_1 and p_2 *expanded patterns* in this paper.

The ì part-ofî relation is used to show the relation between compound and primitive objects. A document is relevant to users if its part-of sections are relevant.

In order to deal with the relationship between classes, we define an identity for each class X, that is, $id(X)$ = {Z| Z is a primitive class, and there is a path from X to Z}, e.g., $id(p_1)$={*pet*, *shop*}. We say class X = class Y if and only if $id(X) = id(Y)$. The following is the basic procedure for automatic ontology extraction:

i) Lexical entry extraction
- Use *tf*idf* (term frequency times inverse document frequency) to get a set of keywords from the training set;
- Select primitive objects (terms) from the set of keywords using the existing background knowledge, where each term denotes a group of keywords, e.g., term ì petî may include {pet, dog, cat};

ii) Determine compound objects (expanded patterns);
iii) Decide the *id* (a set of terms) for all compound objects;
iv) Generate a graph representation as one shown in Figure 1.

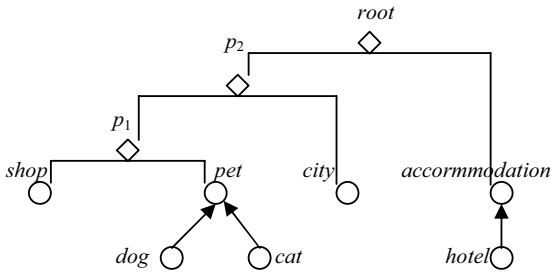

Figure 1 A backbone of the ontology, where arrows denote the ì is-aî relation, and diamond arrows denote the ì part-ofî relation.

3. Reasoning on Ontology

In this section we discuss how to use the discovered knowledge on the ontology to answer what users want. We assume that Ω is the ontology, which includes two set of classes: primitive and compound. We also assume that every testing object o can be represented as a subset $id(o) \subseteq \Theta$.

3.1 Deploying Patterns

Let PL={(p_1, N_1), (p_2, N_2), Ö , (p_n, N_n) } be the set of expanded patterns on the ontology Ω, where p_i (1>*i*>*n*) denote the compound objects, and N_i (1>*i*>*n*) denote the number of appearance of the similar objects. For example, we have PL = {(p_1, 2), (p_2, 1)} as shown in Figure 1, where 2 means two regular patterns map to p_1 (e.g., $d_1 \rightarrow p_1$ and $d_2 \rightarrow p_1$) and 1 means one regular pattern maps to p_2 (e.g., $d_3 \rightarrow p_2$). From PL we can get a *support* function, which satisfies:

$support : P \rightarrow [0,1]$, such that

$$support(p_i) = \frac{N_i}{\sum_{(p_j, N_j) \in PL} N_j} \qquad (1)$$

where $P = \{p \mid (p, N) \in PL\}$.

We can use the following set-valued mapping (see [16] [17]) to illustrate the knowledge implied in *PL*:

$$\Gamma: P \to 2^{\Theta} - \{\varnothing\}, \ such\ that \quad \Gamma(p_i) = id(p_i) \qquad (2)$$

We call Γ a *deploying function* of *PL*. We can also obtain a probability functions pr_Γ to represent the discovered knowledge on the ontology, which satisfies:

$$pr_\Gamma(\theta) = \sum_{\varnothing \neq A \subseteq \Theta, \theta \in A} \frac{support(\{p \in P \mid \Gamma(p) = A\})}{|A|} \qquad (3)$$

for all $\theta \in \Theta$, where $support(X) = \sum_{x \in X} support(x)$ for all $X \subseteq \Theta$. This function is also called *pignistic* probability in [31].

3.2 Decision Rules

Given a testing object *o*, our basic assumption is that *o* is relevant if and only if $\exists p \in P \Rightarrow id(p) \subseteq id(o)$. The set of all objects *o* in the testing set such that $id(p) \subseteq id(o)$ is called the *covering set* for *p* and denoted as [*p*]. The positive region (*POS*) is the union of all covering sets for all $p \in P$.

Except *POS* there are many boundary objects *o* such that $\exists\ p \in P \Rightarrow id(p) \cap id(o) \neq \varnothing$. The set of all objects *o* in the testing set such that $\exists\ p \in P \Rightarrow id(p) \cap id(o) \neq \varnothing$ is called the boundary region (*BND*). Also, the set of all objects *o* in the testing set such that $\forall\ p \in P \Rightarrow id(p) \cap id(o) = \varnothing$ is called the negative region (*NEG*). Given an object *o*, the *decision rules* can be determined naturally as follows (see [16]):

$$\frac{\exists\ p \in P \Rightarrow id(p) \subseteq id(o)}{o \in POS}, \ \frac{\exists\ p \in P \Rightarrow id(p) \cap id(o) \neq \varnothing}{o \in BND} \text{ and } \frac{\forall\ p \in P \Rightarrow id(p) \cap id(o) = \varnothing}{o \in NEG}.$$

3.3 Filtering Algorithm

It must be more interesting to determine thresholds from the discovered knowledge rather than by users. The following is the idea for determining the thresholds in ontology-based Web mining (see [16]).

We have proved that probability function pr_Γ on Θ (see Equation (3)) has the following property:

$$\sum_{t \in id(o)} pr_\Gamma(t) \geq min_{p \in P}\{\sum_{t \in \Gamma(p)} pr_\Gamma(t)\} \qquad (4)$$

for all $o \in POS$. This result shows that the right side formula of Equation (4) can be used as a threshold since we can retrieve all positive documents (i.e., *POS*) using this *threshold*. In [16] a novel filtering algorithm is presented using the above decision rules and the threshold, which finds all of terms occurred in the document, and then sum these termsí probabilities as the probable relevance of the document.

To improve the effectiveness of filtering, in this paper we consider term frequency distributions for the discovered patterns. Now the deploying function Γ in Equation (2) can be extended to the following mapping β, which satisfies:

$$\beta: P \to 2^{\Theta \times I} - \{\varnothing\} \text{ such that } \varnothing \neq \beta(p_i) \subseteq \Theta \times I, \text{ and}$$
$$\Gamma(p_i) = \{fst \mid (fst, snd) \in \beta(p_i)\}$$

for all patterns $p_i \in P$, where *I* is the set of integers, *fst* denotes a term and *snd* denotes the termís frequency in the pattern. For example, we have:

$\beta(p_1)$ = {(*pet*, 4+5), (*shop*, 6+15)}
= {(*pet*, 9), (*shop*, 21)},
$\beta(p_2)$ = {(*pet*, 3), (*shop*, 7), (*city*, 10)}

(See Figure 1, also notice: $d_1 \to p_1$, $d_2 \to p_1$ and $d_3 \to p_2$). Moreover, we can normalize mapping β into an extended random set (see [14] [18]), which satisfies:

$$\beta: P \to 2^{\Theta \times [0,1]} - \{\varnothing\} \text{ such that } \sum_{(fst, snd) \in \beta(p_i)} snd = 1$$
$$\text{and } \Gamma(p_i) = \{fst \mid (fst, snd) \in \beta(p_i)\} \qquad (5)$$

for all patterns $p_i \in P$. We call β a frequency distribution of Γ.

Also, we can obtain a probability function pr_β such that

$$pr_\beta(\theta) = \sum_{p_i \in P, (\theta, snd) \in \beta(p_i)} support(p_i) \times snd \qquad (6)$$

for all $\theta \in \Theta$ (see [14]). In addition, we can prove that Equation (4) is hold if we replace pr_Γ with pr_β.

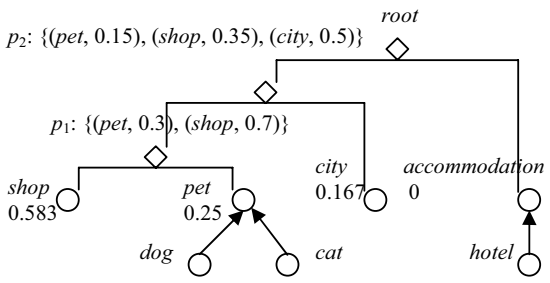

Figure 2 Automatic ontology extraction.

Using the above examples, we have
PL = {(p_1, 2), (p_2, 1)};
Γ(p_1) = {*pet*, *shop*} and Γ(p_2) = {*pet*, *shop*, *city*};
support(p_1) = 2/3 and *support*(p_2) = 1/3; and
$\beta(p_1)$ = {(*pet*, 9), (*shop*, 21)} and
$\beta(p_2)$ = {(*pet*, 3), (*shop*, 7), (*city*, 10)}.
After normalizing β we have the following frequency distribution, which satisfies:

$\beta(p_1)$ = {(*pet*, 9÷30), (*shop*, 21÷30)}
= {(*pet*, 0.3), (*shop*, 0.7)}, and
$\beta(p_2)$ = {(*pet*, 3÷20), (*shop*, 7÷20), (*city*, 10÷20)}
= {(*pet*, 0.15), (*shop*, 0.35), (*city*, 0.5)}.

At last we can obtain the probability function pr_β, which satisfies:

$pr_\beta(shop) = (2/3) * 0.7 + (1/3) * 0.35 = 0.583$

$pr_\beta(pet) = (2/3) * 0.3 + (1/3) * 0.15 = 0.25$

$pr_\beta(city) = (1/3) * 0.5 = 0.167$

$pr_\beta(accommodation) = 0$

The ontology shown in Figure 1 now is replaced by the ontology shown in Figure 2. According to the above analysis, a filtering algorithm is presented as follows.

Algorithm 3.1 Ontology-based filtering.
Input: A *support* function on *PL*; Θ, a set of primitive objects; and β, a frequency distribution of Γ.
Output: *REL*, a set of retrieved object.
Method:
1. let $REL=\varnothing$; //initialization
   for each $\theta \in \Theta$ // initialize the probability function
   let $pr(\theta)= 0$;
2. for each $(p, N) \in PL$ // calculate the probability pr_β
   for each $(\theta, snd) \in \beta(p)$
   $pr(\theta)=pr(\theta)+support(p)\times snd$;
3. let $threshold= min_{p \in P}\{\sum_{t\in\Gamma(p)} pr(t)\}$;
4. for each testing object o {
4.1 let $weight= \sum_{t\in id(o)} pr(t)$;
4.2 if ($weight \geq threshold$)
   $REL = REL \cup \{o\};\}$

The knowledge distributed on the classes of the ontology is transferred to a probability on Θ in the filtering algorithm firstly. It then determines a *threshold* in *step 3*. In *step 4*, it calculates the weight for each testing object in *step 4.1* using the probability and then makes a decision according to the *threshold* in *step 4.2*. The output is *REL* that includes the retrieved objects.

4. Capturing Evolving Patterns

In this section we discuss how to maintain the discovered knowledge on the ontology. The patterns we used in the above sections are *positive patterns* which are what users need (or called relevant data). It is possible to obtain a lot of irrelevant data which are not what users need because usually there is only a very small amount of relevant data. To deduce the burden of on-line training, the ontology-based Web mining model uses positive patterns first to obtain knowledge for user profiles, and then it adopts part of irrelevant data to update the knowledge.

In the beginning we believe every positive case has the same importance, so we have defined a support function (see Equation (1)) to show our belief for all positive patterns. It is easy to update the knowledge for user profiles if we obtain new positive patterns. We only need to add the new positive patterns in *PL*, and then we ì update the *support* functionî using Equation (1) for all $p \in P$. The probability function in *Algorithm 3.1* is re-written when the support function is updated. This treatment is still followed by our initial assumption, that is, every positive pattern has the same importance.

Unfortunately most Web mining systems make some incorrect decisions. The main reason is that we have not obtained correct positive patterns. For example, we may use the same set of keywords to represent a relevant document and a non-relevant document as well. Fortunately, we can use the incorrect decisions to remove some bad positive patterns, or even refine some positive patterns.

We call a testing object *negative pattern* if it is marked in relevance by the Web mining system but it is actually an irrelevant data to users. When we obtain a negative pattern, *np*, we will first check which positive patterns have been used to cause such error. We call these positive patterns *offenders* of *np*. The set of offenders of *np* can be determined by the following equation:

$$\Delta_{np} = \{p_i \mid (p_i, N_i) \in PL, \Gamma(p_i) \cap id(np) \neq \varnothing\}.$$

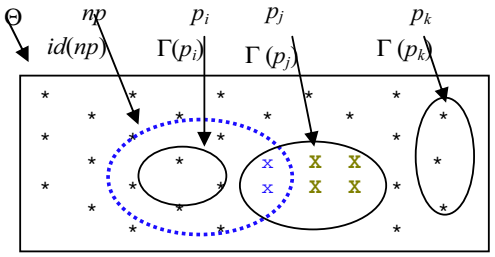

Figure 3 Negative pattern and its offenders

Figure 3 shows an example of a negative pattern *np* and its offenders, where there are three positive patterns: p_i, p_j and p_k. We have $\Delta_{np} = \{p_i, p_j\}$ since $\Gamma(p_i) \cap id(np) \neq \varnothing$, $\Gamma(p_j) \cap id(np) \neq \varnothing$ but $\Gamma(p_k) \cap id(np) = \varnothing$.

There are two kinds of offenders: total conflict offenders whose *ids* are subsets of *id*(*np*), and partial conflict offenders whose *ids* are not subsets of *id*(*np*) but join with *id*(*np*). For instance, p_i as shown in Figure 3 is a total conflict offender since $\Gamma(p_i) \subseteq id(np)$; and p_j as shown in Figure 3 is a partial conflict offender since $\Gamma(p_j) \not\subset id(np)$ and $\Gamma(p_j) \cap id(np) \neq \varnothing$.

In the ontology-based Web mining model we first remove all total conflict offenders from *PL*, and then ì update the *support* functionî. For each partial conflict offender, we determine the offering of the offender from the joint part of its *id* and *id*(*np*) (for convenience, we call this part the lower case part) using the following equation:

$$offering: \Delta_{np} \rightarrow [0,1], \text{ such that}$$

$$offering(p) = \sum_{(fst,snd)\in\beta(p), fst\in id(np)} snd$$

for all partial conflict offenders $p \in \Delta_{np}$.

We also reshuffle the partial conflict offenderís frequency distribution by shifting part of the offering to the rest of part of its *id* (we call this part the upper case part). For example, given a partial conflict offender p_j as shown in Figure 3, the $\frac{1}{\mu}$ of offering from its two lower case x elements will be shifted to its four upper case **X** elements, where μ is an experimental coefficient and $\mu \geq 2$.

The details of the above idea can be found in *Algorithm 4.1*. This algorithm updates the discovered knowledge. It may simply add the new positive patterns, remove some existing positive patterns or even reshuffle frequency distributions of some existing positive patterns.

Algorithm 4.1 Capturing evolving patterns.
Input: (np, N_{np}), a new expanded pattern; and $id(np)$.
Output: Updated *PL*, *support* function and β.
Method:

1. if (*np* is relevant) { // for a new positive pattern
   $PL = PL \cup \{(np, N_{np})\}$; //simply add it to *PL*
   update the *support* function;}
2. else { //get the set of offenders first
   let $\Delta_{np} = \{p_i \mid (p_i, N_i) \in PL, \Gamma(p_i) \cap id(np) \neq \varnothing\}$;
   removed = false; //test if remove any patterns
   for each $p \in \Delta_{np}$
      if ($\Gamma(p) \subseteq id(np)$) { //remove patterns
         $PL = PL$ ñ {(*p*, _)}; //_ match any values
         removed = true;}
      else { //reshuffle frequency distribution
         $offering = (\frac{1}{\mu}) \times \sum_{(fst,snd)\in\beta(p), fst\in id(np)} snd$;
         $base = \sum_{(fst,snd)\in\beta(p), fst\notin id(np)} snd$;
         for each $(fst,snd) \in \beta(p)$
            if ($fst \in id(np)$) // for lower case part
               $snd = (\frac{1}{\mu}) \times snd$;
            else //for upper case part
               $snd = snd + (offering \times snd) \div base$;}
   if (*removed*) //update *support* function
      update the *support* function;}

The variable *base* is never zero and we can easily verify that *Algorithm 4.1* is correct. It is obvious the time complexity of *Algorithm 4.1* is determined by *step 2*. The time complexity of obtaining the set of offenders is $O(|P| \times (|\Theta|)^2)$, where $|P|$ is the number of positive patterns, and $|\Theta|$ is the number of primitive objects. It also takes $O(|\Delta_{np}| \times ((|\Theta|)^2 + |\Theta|))$ in the for loop of *step 2*. Because $|\Delta_{np}| \leq |P|$, the time complexity of *Algorithm 4.1* is $O(|P| \times (|\Theta|)^2)$.

The reshuffle operation does not change $support(p) \times \sum_{(fst,snd)\in\beta(p)} snd$, the belief for each $p \in P$, that means the *threshold* is not changed, and we can retrieve all positive documents (i.e., *POS*) using *Algorithm 3.1*. We also can prove that there exists an integer *n* such that after *n* continuous calls of *Algorithm 4.1* we have $weight = \sum_{t\in id(np)} pr_\beta(t)$ less than the *threshold* if *np* is a negative pattern.

5. Testing and Evaluation

In this section, we use experiments to evaluate the algorithms we presented in the previous sections. We use 2002 TREC (Text REtrieval Conference, see http://trec.nist.gov/) data collection (Reuters Corpus data, Volume: 1 ì English language, 1996-08-20 to 1997-08-19î, topics: 101, 102, Ö, 109). The data in our experiments is split into two sets for each topic: a training set (user feedback) and a testing set.

According to the procedure of automatic ontology extraction, we design an ontology-based filtering model *onto* to implement the idea of *Algorithm 3.1*, where we view each relevant document in the training set as a set of term weight pairs (a regular pattern). This model transfers the knowledge on the ontology to a probability function on the set of keywords. For each document in the testing set, *onto* finds all of the keywords that appear in the document, and then sum the termsí probabilities as the probable relevance of the document. We also extend the *onto* filtering model for capturing evolving patterns. We call the new filtering model *evolving*, which first determines the set of negative documents (patterns) and then uses *Algorithm 4.1* to update *PL*, *support* function and β, where μ = 8. The probability function *pr* in *Algorithm 3.1* is also re-written according to updated *support* function and β.

To test filtering models without bias, we do not account the structure and size of documents. We also select the vector space model, Dempster-Shafer (DS) model (see [30] [15]) and probabilistic models (see [9]) as reference models in the experiments.

One of the most effective term-weight techniques in vector space model is called *tf*idf* (term frequency times inverse document frequency). The following equation is used to derive appropriate term weights for a *tf*idf* filtering model for a given document *d*:

$$w_d(t) = tf \times \log(N/n)$$

where *tf* is the number occurrences of term *t* in document *d*; *N* is the total number of documents in the training set; and *n* is the number of documents in the training set that contain term *t*.

Let D_{topic} be the training set of a given *topic*. The following equation is used in the *if*idf* based filtering model to select the top 150 keywords from D_{topic}:

$$w_{topic}(t) = \sum_{d \in D_{topic}, d \text{ is relevant}} w_d(t).$$

The cosine measure is also used to estimate the similarity between a new document *d* in the testing set and a *topic*.

The *DS* filtering model first obtains a mass function on a set of terms using the deploying mapping Γ:

$$m_\Gamma : 2^\Theta \to [0,1]; \text{ such that}$$
$$m_\Gamma(A) = support(\{p \mid p \in P, \Gamma(p) = A\})$$

for all $A \subseteq \Theta$. It then transfers the mass function into a *pignistic* probability (see Equation (3)) to estimate term weights. It uses the total probabilistic formula to evaluate the similarity between document *d* and a topic.

We use the following equation for the basic probabilistic (*prob*) filtering model:

$$w_{prob}(t) = \log(\frac{r}{R} \div \frac{n}{N})$$

where *r* is the number of relevant documents that contain term *t*; *R* is the number of relevant documents in the training set; and *n* and *N* are defined as the same in *tf*idf* model, respectively.

For dealing uncertainties involved in estimating term weights. The *prob* model can be extended as a *prob0.5* filtering model, which estimates the term weights using the following equation:

$$w_{pr}(t) = \log(\frac{r + 0.5}{(R - r) + 0.5} \div \frac{(n - r) + 0.5}{(N - n) - (R - r) + 0.5})$$

where 0.5, an experimental coefficient, is used to account for the uncertainty involved in estimating relevance [9]. Both *prob* and *prob0.5* use the total probabilistic formula to evaluate the similarity between document *d* and a topic.

All filtering models use the same text pre-processing. This procedure includes: removing any non-relevant elements from XML documents; case folding; stemming; stop word removal; selecting a set of 150 keywords.

We use both precision and recall in the experiments, where the *precision* is the fraction of retrieved documents that are relevant to the topic (query), and the *recall* is the fraction of relevant documents that have been retrieved. Therefore, the higher the figures of both precision and recall curves, the more effective the system is.

Given a topic, we obtain two arrays of floats for each filtering model: *x* array and *y* array for the recall and precision, respectively, where the cut-off is 25. Instead of drawing many precision recall curves, we use both *top 25 precision* and *breakeven point*, which are two new methods used in Web mining for testing effectiveness, where a breakeven point is a point in precision and recall curve with the same *x* coordinate and *y* coordinate. The greater both the top 25 precision and the breakeven point, the more effective the filtering model is.

Table 1 and Table 2 are the results of the experiments, which show a comparison between six different filtering methods on nine topics: 101, 102, Ö , and 109. It is no less impressed by the performance of evolving filtering model on both the top 25 precision and breakeven points.

Table 1. Top 25 Precision

Topic	*tf*idf*	*DS*	*prob*	*prob0.5*	*onto*	*evolving*
101	84.0%	92%	100%	96.0%	100%	100.0%
102	92.0%	96%	96%	96.0%	96%	84.0%
103	44.0%	40%	40%	48.0%	52%	40.0%
104	100.0%	96%	96%	100.0%	96%	96.0%
105	52.0%	68%	80%	92.0%	72%	80.0%
106	32.0%	12%	20%	20.0%	8%	32.0%
107	44.0%	36%	20%	24.0%	52%	52.0%
108	40.0%	40%	36%	40.0%	36%	36.0%
109	32.0%	44%	40%	44.0%	40%	68.0%
Avg	57.8%	58.2%	58.7%	62.2%	61.3%	**65.3%**

Both top 25 precision and breakeven points gain a significant increase.

As a result of these experiments we believe that the proposed theoretical framework for capturing evolving patterns is adequate for ontology based Web mining models.

6. Related Work and Conclusions

Web mining can be classified into four categories: Web usage, Web structure, Web user profile and Web content [8] [21] [22] [32].

Table 2. Breakeven point

Topic	*tf*idf*	*DS*	*prob*	*prob0.5*	*onto*	*evolving*
101	0.822805	0.760943	0.796742	0.787655	0.798077	0.798077
102	0.791271	0.78741	0.786335	0.8014	0.788814	0.704539
103	0.401274	0.374165	0.397374	0.42623	0.397374	0.4
104	0.634202	0.645915	0.655487	0.686318	0.684025	0.667196
105	0.421053	0.678572	0.64151	0.681657	0.666868	0.571429
106	0.27907	0.0967742	0.230769	0.230769	0.064516	0.27907
107	0.330508	0.336	0.168	0.189189	0.384615	0.384615
108*						
109	0.305168	0.306769	0.332278	0.386461	0.294708	0.471754
Avg	0.498	0.498	0.501	0.524	0.510	**0.535**

*The breakeven point does not exist.

Earlier work on access usage logs can be found in [4] [25] [35]. The primitive object of Web usage mining is the discovery of Web server access patterns. It can obtain some interesting patterns about user behaviors from usage logs [36]. Web structure mining is the discovery of hypertext/linking structure patterns [4] [13] [29]. Web user profile mining is the discovery of concept structures to describe what users want [17] [32]. Web content mining is the discovery of Web document content patterns. The goal of content mining is the discovery of useful and interesting patterns [3].

Association mining has been used in Web text mining, which refers to the process of searching through unstructured data on the Web and deriving meaning from it [6][7][12]. The main purposes of text mining were association discovery, trends discovery, and event discovery [3]. The association between a set of keywords and a predefined category (e.g., a term) can be described as an association rule. The trends discovery means the discovery of phrases, a sort of sequence association rules. The event discovery is the identification of stories in continuous news streams. Usually clustering based mining techniques can be used for such a purpose. It was also necessary to combine association rule mining with the existing taxonomies in order to determine useful patterns [2] [7].

To compare with probabilistic models, data mining-based Web mining models did not use term independent assumption [1] [19]. Also, Web mining models tried to discover some unexpected useful data [3] [20]. It is no doubt that the mining procedure can provide many discovered patterns (including both expect and unexpected patterns) from the Web data. However a very difficult problem is that some discovered patterns contain uncertainties while we use the discovered patterns to solve problems.

One possible solution for solving this problem was to research the similarity between patterns in order to discover the real useful patterns [26] [33]. Another solution was to interpret the meaning of patterns using mathematical models. If databases were represented as relational databases, and the users could determine the premises and conclusions of rules; rough set [10] [28] and extended random set [14] [18] based decision rules were used to show the meaning of patterns.

But in many cases we do not know how to differentiate premises and conclusions in the discovered patterns. We may only obtain regular patterns (frequent itemsets) with the formats of sets of terms or sets of term-weight pairs. In this paper, we present a theoretical framework for use and maintenance of regular patterns. To compare with the other approaches, we use an ontology to represent the discovered patterns. The advantage is that we can reason and revise discovered patterns on the ontology in order to eliminate uncertainties. We have verified that our approach not only gains a better performance on both precision and recall, it also decreases the burden of on-line training since the mining model only needs relevant data and part of irrelevant data in the training set.

REFERENCES

[1] M. L. Antonie and O. R. Zaiane, Text document categorization by term association, 2nd *IEEE International Conference on Data Mining*, Japan, 2002, 19-26.

[2] Y. Bi, T. Anderson, S. McClean, A rough set model with ontologies for discovering maximal association rules in document collection, *Knowledge-Based Systems*, 2003, **16:** 243-251.

[3] G. Chang, M.J. Healey, J. A. M. McHugh, and J. T. L. Wang, *Mining the World Wide Web: an information search approach*, Kluwer Academic Publishers, 2001.

[4] M. Chen, J. Park, and P. Yu, Data mining for path traversal patterns in a Web environment, *16th International Conference on Distributed Computing Systems*, 1996, Hong Kong, 385-392.

[5] R. Feldman and H. Hirsh, Mining associations in text in presence of background knowledge, *2nd ACM SIGKDD*

international Conference on Knowledge Discovery and Data Mining, 1996, 343-346.

[6] R. Feldman, I. Dagen, and H. Hirsh, Mining text using keywords distributions, *Journal of Intelligent Information Systems*, 1998, **10(3):** 281-300.

[7] R. Feldman, M. Fresko, Y. Kinar, Y. Lindell, O. Liphstat, M. Rajman, Y. Schler, and O. Zamir, Text mining at the term level, Lecture Notes in AI 1510: *Principle of data mining and knowledge discovery*, Springer-Verlag, 1998, 65-73.

[8] M. N. Garofalakis, R. Rastogi, S. Seshadri, and K. Shim, Data mining and the Web: past, present and future, *WIDM99 conference*, 1999, Kansas City, Missouri, 43-47.

[9] D. A. Grossman and O. Frieder, *Information retrieval algorithms and heuristics*, Kluwer Academic Publishers, Boston, 1998.

[10] J. W. Guan, D. A. Bell, D. Y. Liu, The rough set approach to association rules, 3rd *IEEE International Conference on Data Mining*, 2003, Melbourne, Florida, USA, 529-532.

[11] K. Hoashi, K. Matsumoto, N. Inoue, and K. Hashimoto, Document filtering method using non-relevant information profile*, 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2000, 176-183.

[12] J. D. Holt and S. M. Chung, Multipass algorithms for mining association rules in text databases, *Knowledge and Information Systems*, 2001, **3:** 168-183.

[13] J. M. Kleinberg, Authoritative sources in a hyperlinked environment, *Annual ACM-SIAM Symposium on Discrete Algorithms*, 1998, New York, 668-677.

[14] Y. Li, Extended random sets for knowledge discovery in information systems, *9th International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing*, China, 2003, 524-532.

[15] Y. Li, C. Zhang, and J. R. Swan, An information filtering model on the Web and its application in *JobAgent*, *Knowledge-based Systems*, 2000, **13(5):** 285-296.

[16] Y. Li and N. Zhong, Web mining model and its applications on information gathering, to appear in *Knowledge-Based Systems*, 2004.

[17] Y. Li and N. Zhong, Ontology-based Web Mining Model: representations of user profiles, *IEEE/WIC International Conference on Web Intelligence*, 2003, 96-103

[18] Y. Li and N. Zhong, Interpretations of association rules by granular computing, 3rd *IEEE International Conference on Data Mining*, 2003, Melbourne, Florida, USA, 593-596.

[19] B. Liu, Y. Dai, X. Li, W. S. Lee, and P. S. Yu, Building text classifiers using positive and unlabeled examples, 3rd *IEEE International Conference on Data Mining*, Melbourne, Florida, USA, 2003, pp.179-186.

[20] B. Liu, Y. Ma, and Philip S. Yu, Discovery unexpected information from your competitorís Web sites, *7th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining*, 2001.

[21] Z. Y. Lu, Y.Y. Yao, and N. Zhong, Web log mining, in: N. Zhong, J. Liu and Y. Y. Yao (eds.), ì*Web Intelligence*î, Springer-Verlag, 2003, 174-194.

[22] S.M. Madria, S.S. Bhowmick, W.K. Ng, and E.-P. Lim, Research issues in Web data mining, *1st International Conference on Data Warehousing and Knowledge Discovery*, 1999, 303-312.

[23] A. Maedche, *Ontology learning for the semantic Web*, Kluwer Academic Publishers, 2003.

[24] A. Maedche, V. Pekar, and S. Staab, Ontology learning part one ñ on discovering taxonomic relations from the Web, in: N. Zhong, J. Liu and Y. Y. Yao (eds.), ì*Web Intelligence*î, Springer-Verlag, 2003, 302-319.

[25] H. Mannila and H. Toivonen, Discovering generalized episodes using minimal occurrences, 2nd International Conference on Knowledge Discovery and Data Mining, 1996, Portland, Oregon, 146-151.

[26] H. Motoda, *Active mining*, No. 79 in Frontiers in Artificial Intelligence and Application, IOS Press, 2002.

[27] S. K. Pal and V. Talwar, Web mining in soft computing framework: relevance, state of the art and future directions, *IEEE Transactions on Neural Networks,* 2002, **13(5):** 1163-1177.

[28] Z. Pawlak, Rough sets and intelligent data analysis, *International J. of Information and Sciences*, 2002, **147:** 1-12.

[29] M. Perkowitz, O. Etzioni, Adaptive Web sites, *Communications of the ACM*, 2000, **43(8):** 152-158.

[30] S. Schocken and R. A. Hummel, On the use of the Dempster Shafer model in information indexing and retrieval applications, *Int. J. Man-Machine Studies*, 1993, **39:** 843-879.

[31] P. Smets and R. Kennes, The transferable belief model, *Artificial Intelligence*, 1994, **66:** 191-234.

[32] J. Srivastava, R. Cooley, M. Deshpande, and P.ñN. Tan, Web usage mining: Discovery and applications of usage pattern from Web data, *SIGKDD Explorations*, 2000, **1(2):**1-12.

[33] S. Tsumoto and S. Hirano, Visualization of ruleís similarity using multidimensional scaling, *3rd IEEE International Conference on Data Mining*, 2003, Melbourne, Florida, USA, 339-346.

[34] C. Welty and N. Guarino, Supporting ontological analysis of taxonomic relationships, *Data & Knowledge Engineering*, 2001, **39:** 51-74.

[35] T. Yan, M. Jacobsen, H. Garcia-Molina, and U. Dayal, From user access patterns to dynamic hypertext linking, 5th International World Wide Web Conference, France, 1996.

[36] Q. Yang, H. Zhang, I. Tian, and Y. Li, Mining Web logs for prediction models in WWW caching and prefetching, *7th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining*, 2001, 473-478.

[37] N. Zhong, Representation and construction of ontologies for web intelligence, *International Journal of Foundation of Computer Science*, 2002, **13(4):** 555-570.