Heuristic Learning of Rules for Information Extraction from Web Documents

Dawei Hu[1,2,3], Huan Li[1,2,3], Tianyong Hao[3], Enhong Chen[1,2], Liu Wenyin[2,3]
[1]Department of Computer Science and Technology, University of Science & Technology of China, Hefei, China
[2]Joint Research Lab of Excellence, CityU-USTC Advanced Research Institute, Suzhou, China
[3]Department of Computer Science, City University of Hong Kong, Hong Kong, China
dwhu@mail.ustc.edu.cn, huanl@mail.ustc.edu.cn, tianyong@cityu.edu.hk, ehchen@ustc.edu.cn, csliuwy@cityu.edu.hk

ABSTRACT
The efficacy of an information extraction system is mostly determined by the quality of the extraction rules. Building these extraction rules is time-consuming and difficult to implement by hand. Hence, we propose a Heuristic Rule Learning (HRL) algorithm which can automatically and efficiently acquire high-quality extraction rules from a user labeled training corpus. Moreover, these extraction rules are maintained at the most suitable generalization level to enhance information extraction efficacy. In HRL, we use a Dynamic tErm eXtraction Technique (DEXT) to construct terms and extraction rules at different generalization levels. The conditional entropy model is used to evaluate the suitability of these different generalization levels of the extraction rules so as to maintain them at a high-quality level. Experimental results show the algorithm's efficacy of acquiring extraction rules at different generalization levels and the efficacy of these extraction rules in the information extraction tasks.

Categories and Subject Descriptors
H.3.3 [**Information Search and Retrieval**]: *retrieval models, search process*.

General Terms
Algorithms, Experimentation

Keywords
Information Extraction (IE), Conditional Entropy, Extraction Rule

1. INTRODUCTION
Over the past years, the explosive growth of information on the web makes it necessary to design a system to help people extract what they want from the infinite web documents. Many researchers in natural language processing started to develop information extraction systems which can extract the specific data items from text documents. The methodologies of these systems are mainly using extraction rules to parse the text documents and extracting the matching data out of these documents. As a result, the efficacies of these systems are mostly determined by the quality of their extraction rules. However, building these extraction rules by hand is tedious and error-prone job. Hence, automatic learning of these extraction rules becomes an important research issue.

In early years, some systems focused on learning single-slot extraction rules which could only extract one information piece of interest from a sentence [1, 4, 9, 11]. When processing a document, these systems must pre-define the relations of these extraction rules; otherwise, they could not organize these extracted isolated information pieces. Consequently, these systems could only process the formal documents with single theme, and when applied to processing the documents on the web, they were too domain-specific since the web documents were so informal that most of them contain several themes. Therefore, some researchers had tried to construct more complicated multi-slot rules. Based on the multi-slot rules, those systems could extract several information pieces from a sentence and correlate them automatically [2, 5, 6, 7, 8, 12, 13, 15]. Obviously, comparing to with single-slot rules, these multi-slot rules were more suitable to extract information from the web. However, these researchers did not give a strategy to control the generalization level of the rules. As being discussed by Hu et al. [3], choosing an appropriate generalization level would greatly enhance the rule's efficacy in extracting knowledge from the documents. Hence, in this paper, we propose a Heuristic Rule Learning (HRL) algorithm which can efficiently acquire high-quality extraction rules automatically from a user labeled training corpus. Moreover, these extraction rules are maintained at the most suitable generalization level to enhance information extraction efficacy.

An extraction rule in HRL is represented as a context-slot sequence in which context patterns (CP) and slots (S) appear alternatively, in which the slot is referred to the string to be extracted and the context pattern is referred to the context information of the slots. The formal definition of our extraction rule will be given in the rest of our paper. On the basis of these rules, we can efficiently locate the information piece corresponding to the desired slot through matching its adjacent

INFOSCALE 2007, June 6-8, Suzhou, China

DOI 10.4108/infoscale.2007.214

context patterns with the text in the document. Moreover, since the slot number of an extraction rule is variable, we can acquire both single- and multi-slot rules using HRL. Additionally, semantic information for the slots can be learned and annotated on the rules to help analyze the semantic class of each extracted information piece and the interrelation of two pieces and organize all these information pieces to construct high-quality knowledge.

To control the generalization level of the extraction rules, we present a frequent term extraction algorithm which can construct the terms whose lengths are variable according to the different implementation situations. On the basis of these frequent terms, we can easily construct the extraction rules at different generalization levels. Moreover, as mentioned before, choosing a suitable generalization level for the extraction rules is an important task to enhance the efficacy of the rules in the information extraction task. Hence, after generating these terms, we use the conditional entropy model to evaluate the sensitivity of each frequent term, and on the basis of this evaluation value, we can estimate whether a term is sensitive to distinguish different types of context information.

We have done two experiments to test the efficacy of our learning algorithm. In the first experiment, we construct five groups of extraction rules based on the terms at different occurrence frequency. These extraction rules are used to extract the birthday for a person. We then apply these five extraction rule groups to extract the birthday for a given person on the test documents from the web to test the quality of our extraction rules. In the second experiment, we construct five groups of extraction rules based on the different sensitive terms and then apply these rules to extracting the birthday for a given person. The empirical results show that the rules which are constructed by the sensitive terms at a more suitable generalization level can perform more accurately.

The remaining part of the paper is organized as follows. Section 2 introduces some related works on extraction rule learning. Section 3 describes our rule representation and the HRL algorithm. We present the experimental results in Section 4 and finally, the concluding remarks and future work in Section 5.

2. RELATED WORK

Efficient learning of the extraction rules is an important issue in information extraction area because the information extraction systems can locate and identify the specific information pieces of the data needed from the documents accurately if a high-quality rule base is available.

Information extraction is coined in the Message Understanding Conferences (MUCs). Earlier approaches [1, 9, 10, 11] are mostly focused on formal documents and the possible rules they can learn are very limited. AutoSlot [9] can only generate single-slot rules from those strings which fit the particularly provided 13 syntax templates. Only syntactic constraints are allowed in their rules. In Crystal [11], both single-slot and multi-slot rules can be generated. To parse the text more accurately, Soderland et al. allow both syntactic and semantic constraints on context patterns and slots. In RAPIER [1], only single-slot rules can be learned. Moreover, as shown in Figure 1, besides the syntactic and semantic constraints, Califf use *1-term* list as the constraints to construct more adaptive rules. Obviously, AutoSlot and RAPIER cannot work efficiently on the web because they use single-slot rules which are insufficient to organize the extracted information pieces from informal documents with several themes. On the other side, all these two systems use *1-term* which has no structure information to construct their context patterns. These approaches loose the restriction of the context patterns and limit the precision of processing documents.

CP_1:	S_1:	CP_2
word: {located, offices, in}	list: length 1	word : ,
list: length 1	syntactic: nnp	tag : ,

Figure 1. The rule extracted by RAPIER.

To extract and organize the information pieces from the informal documents on the web, we need build more complicated rules which can contain several slots and semantic relations of these slots. Therefore, some researchers focus on generating the multi-slot rules which can be used to extract several correlated information simultaneously. In a multi-slot rule, several slots exist and a context pattern exists between each two adjacent slots. Moreover, the semantic relation of each pair of two slots is defined in the rule. On the basis of these multi-slot rules, the systems [8, 13] can extract several correlated information pieces instead of isolated pieces. Compared with single-slot rules, these multi-slot rules are more efficiently to extract information from the informal document with several themes. At the same time, how to enable context patterns to supply more strict constraints is another important issue. Ravichandran and Zhang used the most specific *n-term* which is a sequence of n words instead of general syntactic and semantic class constraints to construct a hard context patterns as shown in Figure 2 [8, 13]. These hard context patterns enable their systems achieve a high precision but the recall decreases dramatically. Meanwhile, Cui used two sequential 1-term lists to construct a soft context pattern as shown in Figure 3 [2].

CP_1:	S_1:	CP_2:	S_2:	CP_3:
Φ	<NAME>	was born in	<BIRTHDATE>	Φ

Figure 2. The rule extracted by Ravichandran.

CP_1 :		S_1 :		CP_2 :	
NN	0.12	,	0.11	.	0.4	DT$ 0.2
According	0.03	to	0.03	BE$	0.2	VB 0.1
Known	0.09	as	0.20			

Figure 3. The rule extracted by Cui.

Obviously, these learning methods all use the terms in fix length to construct the context pattern. The Dynamic tErm eXtraction Technique in this paper can extract the terms whose lengths are variable to generate the extraction rules in different generalization level according to the different implement situations.

3. THE HRL ALGORITHM

In the following discussion, we firstly describe the representation of our extraction rules and then the Dynamic tErms eXtraction Technique (DEXT). Our DEXT has two parts: 1) Frequent terms extraction algorithm using which we can extract the terms in different lengths and different occurrence frequencies; 2)

Conditional entropy model based evaluation strategy which is used to estimate the sensitivity of the generated terms. At last, we present our HRL algorithm.

3.1 Rule Representation

An extraction rule R is represented by a 3-tuple <T, S, *StC*>. In our extraction rule *R*, $T=\{CP_{-S1}, S_1, CP_{S1-S2}, \ldots, S_n, CP_{Sn-}\}$ is a *sequence* of rule *elements E* which are categorized into *context pattern CP* and *slot S*, in which *CP* and *S* appear alternatively. $|T|$ is denoted as the *length* of *T* and equal to the number of the rule elements in *T*. S_i is denoted as the *i-th* string to be extracted by *R* and $CP_{Si-1-Si}$ is denoted as the context pattern between S_{i-1} and S_i, i.e., $S_i \in T$ for $1 \le i \le n$, $CP_{Si-1-Si} \in T$ for S_{i-1}, $S_i \in T$. Additionally, CP_{-S1} is referred to the context pattern before S_1 and CP_{Sn-} is referred to the context pattern following S_n. Obviously, each slot is embraced by two adjacent context patterns which are applied to matching the text so as to locate the position of their median slot in the documents.

$S=UnS \cup BiS$ is a set of semantic annotations for the slots of the extraction rule *R*. *UnS* is denoted to the semantic class of a slot and we denote $UnS(S_i)$ as the semantic class of S_i. We use WordNet semantic class set as our semantic class set. *BiS* is denoted as the semantic relation of two slots and we denote $BiS(S_i, S_j)$ as the semantic relation between S_i and S_j. We have defined 34 types of binary semantic annotations, as shown in Table 1.

Table 1. Binary semantic annotation types defined.

DateOf	AntonymOf	IsA	SubjectOf
HeightOf	PropertyOf	ShapeOf	CnumberOf
ColorOf	QualityOf	SizeOf	DistanceOf
PartOf	DefinedAs	UsedFor	CapableOf
MannerOf	SynonymOf	TimeOf	SubeventOf
LengthOf	SubclassOf	Nextevent	LocationOf
ObjectOf	SubPropertyOf	WidthOf	MadeOf
TimeCost	SubRelationOf	PriceOf	EffectOf
ReasonOf	UCquantityOf		

StC represents structure constraints for the context patterns of the extraction rule R and we denote $StC(CP_{Si-1-Si})$ as the structure constraints of $CP_{Si-1-Si}$. In our algorithm, *StC* is composed of a context maximum length *StC.Length*, a context type *StC.Type* and two disjunctive lists of terms *StC.NT* and *StC.OT*. *StC.OT* is a list of the terms in which each term may appear in the context pattern while *StC.NT* is a list of terms in which each term must appear in the context pattern. *StC.Type* whose value is T_i in which the subscript *i* is generated by connecting the semantic class of its two adjacent slots. For example, $StC.Type(CP_{Si-1-Si}) = T_{UnS(Si-1)-UnS(Sn)}$.

Figure 4 shows an example of our extraction rule which is used to extract the birthday of some persons. It has five columns, three of them are context patterns and two of them are slots. In this rule, there are two unary semantic annotations and one binary semantic annotation. In which, these two unary semantic annotations are "$UnS(S_1)$= *Person*" which means the extracted information by S_1 belongs to the semantic class "*Person*" in WordNet [14] and "$UnS(S_2)$= *Date*" which means the extracted information by S_2 belongs to the semantic class *"Date"* in WordNet. The only one binary semantic annotation is "$BiS(S_1, S_2)$= *PropertyOf*" which means the information extracted by S_2 is a property of the information extracted by S_1. On the other side, from the structure constraints of CP_{S1-S2}, we know that the context pattern between S_1 and S_2 must consist of "born in" and may have "was" or "am", and its length must be no more than 3 words, and its type is $T_{Person-Date}$. Based on this extraction rule, we can learn from the sentence "John was born in 1994 ." that the date "1994" is a property of the person "John".

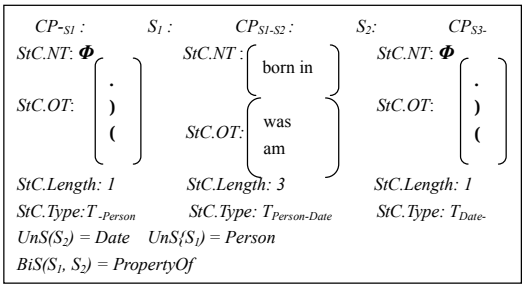

Figure 4. The rule extracted by HRL.

Using this extraction rule we can efficiently locate the information pieces, the semantic class of each piece and the relation of two pieces from the text in the documents. After describing the representation of our extraction rules, we will then provide the Dynamic tErm eXtraction Technique (DEXT).

3.2 Dynamic tErm eXtraction Technique

Compared with those systems which use fix-length terms to construct the structure constraints [8, 13] and the systems which use *1-term* lists to construct the structure constraints [1, 2], we use the variable-length term lists to build our structure constraints. To extract and estimate these variable-length terms, we put forward frequent terms extraction algorithms and conditional entropy based evaluation strategy.

3.2.1 Frequent Terms Extraction Algorithm

Given a context pattern type *T* and its instance list *L*, we use span methodology to extract variable-length terms which frequently occur in *L*. Using this methodology, we firstly expand the frequent *i-term* to obtain the candidate *(i+1)-term*. Then, we count the candidate *(i+1)-term* and eliminate those infrequent ones to obtain frequent *(i+1)*-term.

```
FrequentTermList = Φ
InstanceList = L
CurrentFrequentTermList = Φ
TermList = Initialize (InstanceList)
While (TermList != Φ)
{
    CurrentFrequentTermList =
            RemoveUnFrequentTerms (TermList, θ*|L|)
    If (CurrentFrequentTermList != Φ)
        FrequentTermList.Add(CurrentFrequentTermList)
    TermList = SpanTerms(CurrentFrequentTermList)
    n ++
}
```

Figure 5. Frequent terms extraction algorithm.

Figure 5 shows the detail of our frequent terms extraction algorithm. In our algorithm, there is one parameter θ which is referred to the minimum support that the frequent terms must satisfy and $\theta^*|L|$ means the minimum support count of the frequent terms. To initiate the term set, we firstly take every word occurred in L as a *1-term* and insert them into the term lists. Then we count the occurrence time of each *1-term* in all the instances in L and eliminate the infrequent *1-term* whose occurrence time is less than the minimum support count. After spanning the frequent ones to candidate *2-term* through combining the *1-term* with the word next to it, we update the term list and iterate the above steps until there is no longer generate frequent terms.

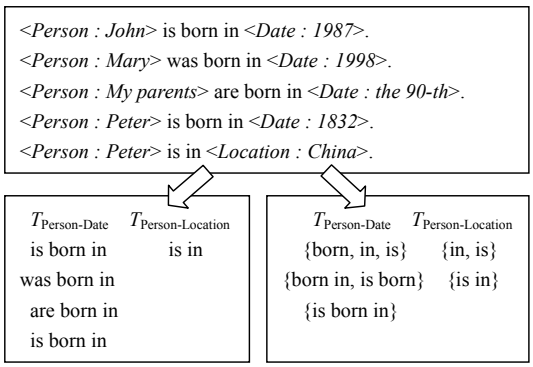

Figure 6. The results of the frequent terms extraction algorithm (θ = 0.5).

Using the above algorithm, we can extract the frequent terms which occur more than $\theta^*|L|$ in L. Figure 6 shows how to extract frequent terms using our algorithm. In this example, θ is set to 0.5. The top figure shows five annotated sentences. The bottom-left figure shows two context pattern types and their instance lists. The bottom-right figure shows the frequent terms we extracted. We can see from the example that, for $T_{Person\text{-}Date}$, we extract three frequent *1-term* which are "born", "in" and "is", two frequent *2-term* which are "born in" and "is born" and one frequent *3-term* which is "is born in" since these six terms reach the minimum support count of 0.5 * 4 = 2. While for $T_{Person\text{-}Location}$, we extract two frequent *1-term* which are "in" and "is" and one frequent *2-term* which is "is in" since these three terms reach the minimum support count of 0.5 * 1 = 0.5.

Having these terms, we then need an evaluation strategy to estimate the quality of these frequent terms, which means we need to estimate whether these terms are representative or sensitive for its corresponding context pattern type. Hence, we use a conditional entropy based evaluation strategy to evaluate it.

3.2.2 Conditional Entropy based Evaluation Strategy

In this section, we describe how to use the conditional entropy model to evaluate the sensitivity of a term for distinguishing different context pattern types. Before we describe our evaluation strategy, we would like to give some definitions, Let *CPL* refer to the context pattern list and CP_i be the *i-th* context pattern in *CPL*, so $CPL = \{CP_1, CP_2, \ldots, CP_n\}$. *CPTL* is referred to the context pattern type list, and the *j-th* context pattern type in *CPTL* is denoted as CPT_i, so $CPTL = \{CPT_1, CPT_2, \ldots, CPT_m\}$. We use equation (1) to calculate the sensitivity of a term T for distinguishing different context pattern types. In equation (1), $p(CP_i, T)$ is referred to the probability of CP_i and T both occurs and $p(CP_i | T_j)$ is referred to the probability of CP_i given T_j. $p(CP_i, T)$ and $p(CP_i | T_j)$ can be calculated using equation (2) and (3), in which $N(T)$ is the occurrence times of T in the entire *CPL*. The larger entropy value H is, the less insensitive the term T is for distinguishing different context pattern types.

$$H(CPTL \mid T) = \sum_{(CPT_i \in CPTL)} p(CPT_i, \quad T) \log \frac{1}{p(CPT_i \mid T)} \quad (1)$$

$$p(CPT_i, \quad T) = \frac{\sum_{(CP_j \in CPL, StC.Type(CP_j)=CPT_i)} \delta(CP_j, T)}{\sum_{(CP_k \in CPL)} \delta(CP_k, T)},$$

$$\delta(CP_j, T) = \begin{cases} 1 & T \in CP_j \\ 0 & T \notin CP_j \end{cases} \quad (2)$$

$$p(CPT_i \mid T) = \frac{\sum_{(CP_j \in CPL, StC.Type(CP_j)=CPT_i)} \delta(CP_j, T)}{N(T)},$$

$$\delta(CP_j, T) = \begin{cases} 1 & T \in CP_j \\ 0 & T \notin CP_j \end{cases} \quad (3)$$

For the example shown in Figure 6, we use the following two equations to calculate the entropy of "is" and "born". Obviously, the term "is" has much larger entropy value than "born", so "born" is more sensitive to distinguish the different context pattern types than the term "is".

$$H(CPTL \mid "is")$$

$$= \sum_{(CP_i \in CPL)} p(CPT_i, \quad "is") \log \frac{1}{p(CPT_i \mid "is")}$$

$$= \frac{2}{3} \log(\frac{3}{2}) + \frac{1}{3} \log(3)$$

$$= 0.866$$

$$H(CPTL \mid "born")$$

$$= \sum_{(CP_i \in CPL)} p(CPT_i, \quad "born") \log \frac{1}{p(CPT_i \mid "born")}$$

$$= 1 \log(1) + 0 = 0$$

3.3 HRL algorithm

In this section, we will describe our Heuristic Rule Learning algorithm (HRL) for information extraction. The input of our algorithm is human labeled training corpus and the output is the extraction rules. Our algorithm has four steps: rule base initialization, frequent term extraction, sensitivity evaluation, and pattern generalization and merging.

3.3.1 Rule Base Initialization

In this step, we initialize our extraction rules base on the basis of a human labeled training corpus. Firstly, we convert each annotated word to an abstract slot and mark the slot with its unary semantic class. Secondly, based on the unary semantic class of

two adjacent slots, we dynamically construct the type of their median context pattern. Then, we collect the instances for each context pattern type and construct its instance list. At last, we initialize rules based on the slot semantic annotations and the context pattern types.

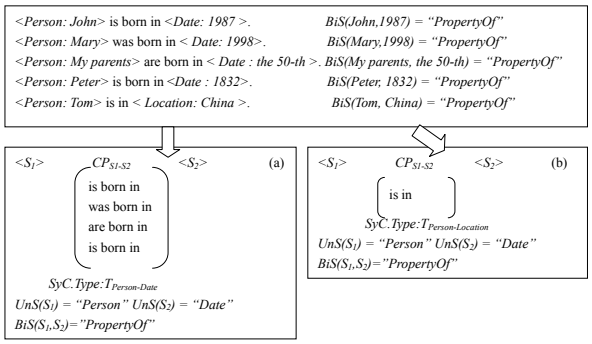

Figure 7. Examples of Initiating Extraction Rule Base.

As shown in Figure 7, for five given training sentences, we firstly generalize the annotated words to the abstract token S and construct the unary semantic class *UnS* for each slot. Then on the basis of the semantic class of each two adjacent slots, we generate two types of context patterns $T_{Person\text{-}Date}$ and $T_{Person\text{-}Location}$. At last, we collect the instances and construct an instance list for each type of context pattern. After converting the context instances to the context pattern tags, we obtain two initial extraction rules Figure 7-(a) and Figure 7-(b).

3.3.2 Frequent Term Extraction

In this step, we use frequent terms extraction method to extract frequent terms in different lengths and construct a frequent term list for each context pattern type in the rule base.

Figure 8 shows the frequent term lists we constructed for $T_{Person\text{-}Date}$ and $T_{Person\text{-}Location}$ if the θ is set to 0.5. The left figure shows the frequent term list for $T_{Person\text{-}Date}$, which contains six terms, "is", "born", "in", "is born", "born in" and "is born in". The right figure shows the frequent term list (FTL) for $T_{Person\text{-}Location}$, which has three terms, "is", "in' and "is in".

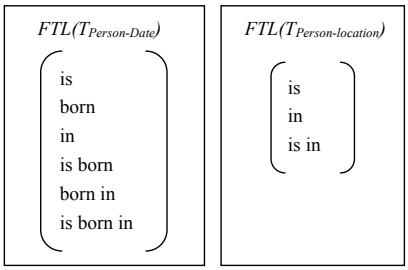

Figure 8. Frequent term lists of $T_{Person\text{-}Date}$ and $T_{Person\text{-}Location}$.

3.3.3 Sensitivity Evaluation

In this step, we use conditional entropy model to evaluate the sensitivity of each term for distinguishing different context pattern types. Then we compare the entropy value with a given threshold H and eliminate those non-sensitive frequent terms whose entropy values are larger than H. Those terms whose entropy values are smaller than H are regarded as sensitive enough to distinguish different context pattern types and will be kept to construct *SyC.NT*. The conditional entropy based evaluation strategy is mentioned in Section 3.2.2 Figure 9 shows the sensitive frequent term list (SFTL) for $T_{person\text{-}date}$ and $T_{Person\text{-}Location}$. In this example, H is set to 0.5. The left graph is the sensitive frequent term list of $T_{Person\text{-}Date}$, and right graph is the sensitive frequent term list of $T_{Person\text{-}Location}$.

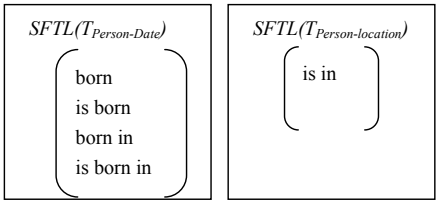

Figure 9. Sensitive frequent term lists.

3.3.4 Pattern Generalization and Merging

In this step, we generalize the context pattern of the initial rules through converting the instance list to two more general term lists *StC.OT* and *StC.NT*. To achieve this, we firstly use all sensitive frequent terms occurred to build *StC.NT* and use all the other words to build *StC.OT* for each instance in the instance list. Then we merge *StC.OT* of all the instances whose *StC.NT* are the same and use the merging result and *StC.NT* instead of the instance list to obtain our final results. For example, Figure 10 shows the operation for the initial rules shown in Figure 7-(a). The Figure 10-(a') is the results of building *StC.OT* and *StC.NT* for each instance in the instance list of $T_{Person\text{-}Date}$. We can see that "was" and "are" are marked as *StC.OT* since these two words are not in $SFTL(T_{Person\text{-}Date})$, while "born in" is marked as *StC.NT* since it is in $SFTL(T_{Person\text{-}Date})$. Then since the *StC.NT* of the second and third instances are equal, we merge *StC.OT* of them and obtain the rule shown in Figure 10-(a''). Moreover, since the *StC.NT* of the first and fourth instances are equal, we merge *StC.OT* of them and obtain the rule shown in Figure 10-(b''). These two rules are our final results.

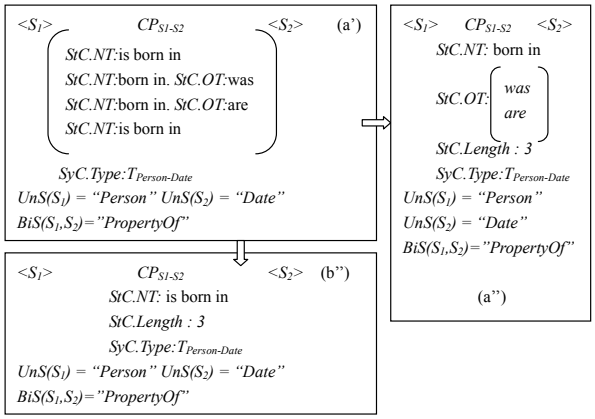

Figure 10. Results of pattern generalization and

4. EXPERIMENTS AND RESULTS

After describing our HRL learning algorithm, experiments on a human labeled training corpus and a test dataset are used to test the efficacy of our HRL algorithm. Our data source is 200 web pages including person description chosen from Google. Then we split the 200 pages into 10 groups and randomly pick one group to annotate. The annotated group is taken as the training corpus and the other 9 groups are taken as test corpus.

<*Person: Mao*> was born on < *Date: December 26, 1893*>
BiS(Mao, December 26, 1893) = PropertyOf
<*Person: Zhou Enlai*> (<*Date: March 5, 1898*> – <*Date: January 8, 1976*>)
BiS(Zhou Enlai, March 5, 1898) = PropertyOf
BiS(Zhou Enlai, January 8, 1976) = PropertyOf
<*Person: Mao Zedong*>, Chinese Communist leader, was born in Hsiang-t'an, China on <*Date: December 26, 1893*>.
BiS(Mao Zedong, December 26, 1893) = PropertyOf

Figure 11. Examples of annotated information pieces.

The Figure 11 shows some annotated examples in our training corpus. We firstly test the efficiency of our frequent terms extraction algorithm. H is set to 0.5 to ensure us to extract the sensitive terms. By varying the value of parameter θ, we extracted three extraction rule sets at different generalization levels from training corpus. Then we apply these different extraction rules to extract the birthday information for a person in the test corpus. We repeat this experiment for 4 times. The size of the different extraction rule sets, the precision and recall of the extraction results are shown in Table 2, in which the precision and recall are calculated as follows:

$$precision = \frac{C_r}{C_a} \quad (4)$$

$$recall = \frac{C_r}{C_{ar}} \quad (5)$$

where, C_a is the number of all the information pieces we extracted, C_r is the number of the correct birthday information we extracted, and C_{ar} is the number of all the birthday information in the test corpus.

From the results, we can see that θ determines the size of extraction rule sets and the recall of the extraction results mostly. When we set θ to 0.03, HRL can learn 11 rules. However, in these rules, any rules are too specific and occur only once in the entire instance lists. These rules contribute little to extracting correct information when be applied to another domain since they are too domain-specific.

Table 2. Illustration the relation of size, precision, recall and θ (H=0.5).

	$\Theta = 0.03$	$\Theta = 0.09$	$\Theta = 0.15$	$\Theta = 0.21$	$\Theta = 0.27$
Size	11	2	2	2	0
Precision	86.70%	100%	100%	100%	0%
Recall	29.55%	18.10%	18.10%	18.10%	0%

Figure 12 shows two examples of the 11 rules. The bottom rule only occurs once and the top rule occurs 15 times. Obviously, the bottom rule is too specific and can hardly be applied to extract the birthday for a new person. The generalization level of the top rule is more suitable since *StC.OT* makes the rule more flexible. When we set θ to 0.09, only 2 rules which occur more frequently are learned by HRL. The bottom rule in Figure 12 cannot be learned this time since the terms in *StC.NT* are not frequent. From the HRL algorithm, we can easily know these 2 rules are covered by the above 11 rules. Comparing the recall of these two rule sets, we can see that these two rules give the more contributions than the other 9 rules in information extraction task. When we set θ to 0.27, no rule is extracted, since there is no such frequently-used extraction rule.

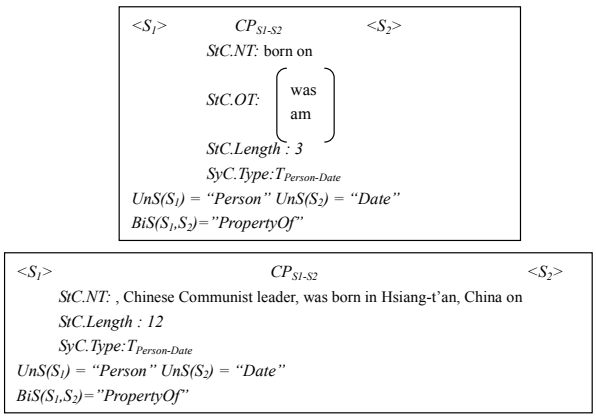

Figure 12. Examples of the rules extracted by HRL.

After testing the impact of θ, we show the impact of H. By varying both θ and H, we extracted different extraction rule sets from the same training corpus. Based on these different rule sets, we extract the birthday information for a person in the same test corpus. Because our learning algorithm is focused on extracting information on the web and there are millions of data to enable us to obtain some results more or less, we focus on the phenomena of precision except for recall. The relation of θ, H and precision of the extraction results is shown in Table 3.

Table 3. Illustration the relation of precision, H and θ.

	$\Theta = 0.03$	$\Theta = 0.09$	$\Theta = 0.15$	$\Theta = 0.21$	$\Theta = 0.27$
$H = 0.5$	86.70%	100%	100%	100%	0%
$H = 1.5$	73.81%	77.78%	77.78%	77.78%	0%
$H = 2.5$	45.23%	60.87%	77.78%	77.78%	44.44%

From the results, we can see that H which reflects the sensitivity of the terms determines the precision of the extraction results. When we set θ to 0.03 and increase H from 0.5 to 2.5, the precision decreases dramatically. However, increasing the θ can counteract some defects by increasing H, since the frequently-used expression of human language has less ambiguity. It is why the precision does not decrease along with increasing the H from 1.5 to 2.5, when we set θ to 0.15. Moreover, it is interesting that when we set θ to 0.27 and increase the H, the precision increases from 0 to 44.44%. The reason is that when we loose the

restriction for the sensitivities of the terms, we can learn some low-quality rules instead of zero high-quality rules. And using these rules, we can obtain some unsatisfied results instead of no results.

In conclusion, the θ determines popularity of the extracted rules while the H determines the quality of the extracted rules. If we choose a suitable H and θ, the HRL algorithm can be an effective and we can use it to acquire high-quality extraction rules.

5. CONCLUSIONS AND FUTURE WORK

We have presented a Heuristic rule learning algorithm named HRL in this paper. In HRL, a Dynamic tErm eXtraction Technique is used to construct terms at different generalization levels. We use the conditional entropy model to evaluate the quality of these different terms so as to use them construct high-quality extraction rules at the most suitable generalization level. Experimental results show its efficacy. That is, it can indeed acquire extraction rules at different generalization levels based on frequent term extraction method, and the entropy value can efficiently estimate the sensitivity of each term. Moreover, we can construct the high-quality extraction rule set based on these sensitive terms. We hope that using the HRL algorithm, we can extract more powerful extraction rule base than those rule bases used by Ravichandran [8] and Cui [2], because in our experiment, we obtain a more accurate results based on our extraction rule set. Compared with the learning algorithm proposed by Zhang [13], our algorithm can learn the extraction rules at different generalization levels depending on the different situations. The extraction rules learned by HRL are more flexible than Zhang's specific rules.

However, there are still some problems of our algorithm and more research works need be done to solve these problems. Firstly, the efficiency of our algorithm is mostly determined by θ and H and we do not know how to choose a suitable value for them. In the future research, we will consider to find a strategy to evaluate θ and H. Secondly, our training corpus is too small and our experiments are focused on extracting birthday information for a person, we plan to use a large training corpus to obtain more powerful extraction rule base which can extract other interesting information as well. Lastly, there are no relations between the terms we learned, so we will use more complex insensitivity evaluation strategy to estimate these terms more accurately.

6. ACKNOWLEDGMENTS

The work described in this paper was fully supported by a grant from City University of Hong Kong (Project No. 7002137), the National Grand Fundamental Research 973 Program of China under Grant No.2003CB317002, and the National Natural Science Foundation of China under Grant No.60573077.

7. REFERENCES

[1] Califf M. E., and R. J. Mooney. *Bottom-up Relational Learning of Pattern Matching Rules for Information Extraction.* Journal of Machine Learning Research, 2003, pp. 177-210.

[2] Cui H., et al. *Generic Soft Pattern Models for Definitional Question Answering.* In Proceedings of SIGIR-05, 2005.

[3] Hu D., et al. *SIIPU*S: A Semantic Pattern Learning Algorithm.* In Proceedings of the second international conference on Semantics, Knowledge and Grid (SKG2006), 2006.

[4] Kim, J., and Moldovan, D. *Acquisition of Linguistic Patterns for Knowledge-based Information Extraction.* In IEEE Transactions on Knowledge and Data Engineering, 1995, pp. 713-724.

[5] Kwok C., et al. *Scaling Question Answering to the Web.* In Proceedings of the World Wide Web Conference-10 (WWW'10), 2001, pp. 150-161.

[6] Mark A. G., and Horacio S. *A Pattern Based Approach to Answering Factoid, List and Definition Questions.* In Proceedings of the 7th RIAO Conference (RIAO 2004). Avignon, France, April 27, 2004.

[7] Moldovan D., et al. *LASSO: A Tool for Surfing the Answer Net.* In Proceedings of TREC-8, 1999.

[8] Ravichandran D., and Eduard Hovy. *Learning Surface Text Patterns for a Question Answering System.* In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, 2002, pp. 41–47.

[9] Riloff, E. *Automatically Constructing a Dictionary for Information Extraction Tasks.* In Proceedings of the 11th National Conference on Artificial Intelligence (AAAI-93), 1993.

[10] Soderland, S. *Leaning Information Extraction Rules for Semi-structured and Free Text.* Journal of Machine Learning, 1998.

[11] Soderland S. et al. *Crystal: Inducing a Conceptual Dictionary.* In Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI -95), 1995, pp. 1314 -1319.

[12] Soubbotin M. M., and Soubbotin S. M. *Patterns and Potential Answer Expressions as Clues to the Right Answers.* In Proceedings of TREC-10, 2001.

[13] Zhang D., and Lee W. S. *Web based Pattern Mining and Matching Approach to Question Answering.* In Proceedings of TREC-10, 2001.

[14] WordNet, http://wordnet.princeton.edu/

[15] BuyAns, http://www.buyans.com/