Cascading Use of Soft and Hard Matching Pattern Rules for Weakly Supervised Information Extraction

Jing Xiao
School of Computing, National University of Singapore, 117543
xiaojing@comp.nus.edu.sg

Tat-Seng Chua
School of Computing, National University of Singapore, 117543
chuats@comp.nus.edu.sg

Hang Cui
School of Computing, National University of Singapore, 117543
cuihang@comp.nus.edu.sg

Abstract

Current rule induction techniques based on hard matching (*i.e.,* strict slot-by-slot matching) tend to fare poorly in extracting information from natural language texts, which often exhibit great variations. The reason is that hard matching techniques result in relatively high precision but low recall. To tackle this problem, we take advantage of the newly proposed soft pattern rules which offer high recall through the use of probabilistic matching. We propose a bootstrapping framework in which soft and hard matching pattern rules are combined in a cascading manner to realize a weakly supervised rule induction scheme. The system starts with a small set of hand-tagged instances. At each iteration, we first generate soft pattern rules and utilize them to tag new training instances automatically. We then apply hard pattern rule induction on the overall tagged data to generate more precise rules, which are used to tag the data again. The process can be repeated until satisfactory results are obtained. Our experimental results show that our bootstrapping scheme with two cascaded learners approaches the performance of a fully supervised information extraction system while using much fewer hand-tagged instances.

1 Introduction

Information Extraction (IE) aims to extract specific information items of interest from free or semi-structured texts, and pattern rule induction is one of the most common techniques for IE tasks (Muslea, 1999). There has been much work in learning extraction pattern rules from tagged data, *e.g.,* AutoSlog-TS (Riloff, 1996), WHISK (Soderland, 1999) and LP^2 (Ciravegna, 2001). In a typical IE system, generalized pattern rules are usually represented as regular expressions and matched against test instances through exact matching for each slot, which we call *hard matching*. Utilizing hard matching pattern rules could obtain precise results from test instances. However, the approach is problematic in dealing with natural language text, such as news articles, which often exhibits great variations in both lexical and syntactic constructions. For instance, in the terrorism domain, given a common rule "<victim> be kidnapped by …", hard matching pattern rules cannot pick up the instance "<victim> , kidnapped by …" due to the mismatch in only one token. Such hard matching techniques often result in low recall. To achieve flexibility in pattern matching for natural language texts, soft matching pattern rules have been proposed for question answering (Cui, *et al.*, 2004). Soft pattern rules match test instances using a probabilistic model to better accommodate variations in expressions. However, differing from the question answering problem, the IE task needs to precisely locate the boundaries of the extracted slots. As such, soft pattern rules may not meet the precision requirement of the task.

In this paper, we aim to minimize the number of hand-tagged training instances needed to start the learning process by adopting a bootstrapping strategy such as that proposed in Riloff and Jones (1999). In contrast to the existing work, we propose a weakly supervised IE framework which takes advantages of both soft and hard matching pattern rules in both the training and test phases. Starting with only a small set of hand-tagged training instances, we first generate a set of soft pattern rules and utilize them to tag more training instances. Next, we apply a hard matching pattern rule induction algorithm, GRID (Xiao, *et al.,* 2003), over both manually and automatically tagged instances to generalize precise hard-matching rules. These hard pattern rules are utilized to tag training instances for soft pattern rule generation in the next iteration. The process runs iteratively till the termination criteria are met. At the end of the training process, we obtain two sets of pattern rules, namely the hard and soft pattern rules. During the test phase, both sets of pattern rules are used in a cascaded way, with hard pattern rules followed by soft pattern rules, to extract target slots from new documents. We have conducted two experiments on both semi-structured and free texts to demonstrate the effectiveness of our method. The experimental results show that the bootstrapping scheme with two cascaded pattern rule learners could achieve a performance close to that obtained by fully supervised learning while using only 5~10% of the hand-tagged data.

The main contribution of our work is in incorporating soft matching pattern rules in the bootstrapping framework. Rooted in instance-based learning, soft pattern rules are more appropriate in dealing with sparse data (Cui, *et al.*, 2004), and thus can be learned from a relatively small number of training instances to start the bootstrapping process. Moreover, in test phase, soft pattern rules are expected to cover more unseen instances, which are likely to be missed by hard-matching rules, with its flexible matching mechanism.

The rest of the paper is organized as follows. Section 2 presents the design of our system. Section 3 describes the details of data preparation, soft pattern matching, hard pattern rule induction and the application of the two pattern rules on new test instances. Section 4 presents the experimental evaluation. We review other work in Section 5 and conclude the paper in Section 6.

2 System Design

Figure 1 shows the overall system architecture of our IE system. The training phase of the system is carried out as follows:

(a) We take a small set of hand-tagged instances (*seed instances*) provided by the user.

(b) We generate soft pattern rules using the seed instances, and denote the soft pattern rules as SP_i.

(c) We apply the learned soft pattern rules (SP_i) to automatically tag unannotated data. We employ a simple cut-off strategy that keeps only the highly-ranked instances by the soft pattern rules.

(d) We generate hard pattern rules using GRID over the automatically tagged instances and seed instances. The resulting hard pattern rules are denoted as HP_i.

(e) If the termination condition is satisfied, the process ends with a set of learned soft and hard pattern rules. Otherwise, the hard pattern rules HP_i are used to tag the training data again. We start a new round of training from Step (b) using the newly tagged training instances and seed instances.

In the test phase, we apply both the hard and soft pattern rules to match against test instances. Specifically, soft matching pattern rules would assign a probabilistic score to an instance that is not matched by any of the hard matching pattern rules. Only those fields that are matched by the hard pattern rules or have high scores in soft pattern matching will be extracted.

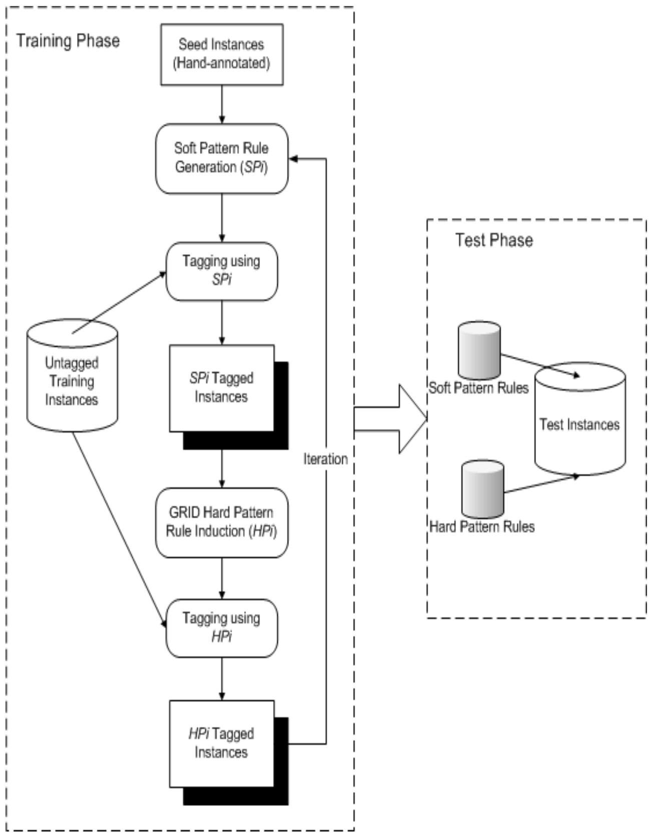

Figure 1: Architecture of our IE system

3 Soft and Hard Pattern Rule Learning

3.1 Data Preparation

Before pattern rule learning commences, we pre-process the training and test instances by using a natural language chunker [1] to perform part-of-speech (PoS) tagging and chunking. We also use a rule-based named entity tagger (Chua and Liu, 2002) to capture semantic entities. Given a tagged instance, we consider the left and right k chunks around the tagged slot as the context:

$<c_{-k}>\ldots<c_{-2}><c_{-1}>tagged\_slot<c_{+1}><c_{+2}>\ldots<c_{+k}>$ (1)

Here $<c_i>$ {i=-k to +k} represents the contextual chunks (or slots) of the tagged slot, where k is the number of contextual slots considered. $<c_i>$ can be of various feature types, namely words, punctuations, chunking tags like verb and noun phrases, or semantic classes. We perform selective substitution to generalize the specific terms in each slot so as to make the learned pattern rules general enough to be applied to other instances. Table 1 shows the substitution heuristics employed in our system with examples.

Figure 2 gives five examples of original training instances for “starting time” in the seminar announcement domain. We substitute the more

[1] We use NLProcessor, a commercial parser from Infogistics Ltd. http://www.infogistics.com/.

general syntactic or semantic classes for the lexical tokens according to the heuristics in Table 1.

Tokens	Substitution	Examples
9 types of named entities	NP_Person, NP_Location, NP_Organization, NP_Date, NP_Day, NP_Time, NP_Percentage, NP_Money, NP_Number.	"Friday"→NP_Day "Feb.27"→NP_Date
Noun Phrase	NP_HeadNoun	"the seminar" →NP_seminar
Verb Phrase (passive or active)	VPpas_RootVerb, VPact_RootVerb	"will speak" →VPact_speak, "will be held" →VPpas_hold
Preposition Phrase	PP	"in civilian clothes" → PP
Adjectival and adverbial modifiers	*To be deleted*	
All other words and punctuations	*No substitution*	"Time", "at", "by", etc. are unchanged.

Table 1: Substitution heuristics

```
(1) Original instances for slot <stime>:
    Time : <stime> 2:30 PM </stime>
    … will be at <stime> 3 pm </stime> …
    …Friday, February 17 <stime> 12:00pm </stime> - 1:00pm
    … will be at <stime> 4pm </stime> , Monday, Feb. 27 …
    Time: <stime> 12:00 PM </stime> - 1:30 PM
(2) Substituted instances:
    Time : <stime> NP_Time </stime>
    VPact_be at <stime> NP_Time </stime>
    NP_Day , NP_Date <stime> NP_Time </stime> - NP_Time
    VPact_be at <stime> NP_Time </stime> , NP_Day , NP_Date
    Time : <stime> NP_Time </stime> - NP_Time
```

Figure 2: Illustration of generalizing instances

3.2 Soft Matching Pattern Rules

Soft pattern rules have been successfully applied to text mining (Nahm and Mooney, 2001) and question answering (Cui, *et al.*, 2004). We employ a variation of the soft pattern rules generation and matching method presented in Cui, *et al.* (2004). We expect soft pattern rules to offer higher coverage in matching against a variety of instances in both the training and test phases.

For each type of tagged slot ($Slot_0$) such as *stime* in Figure 2, we accumulate all the tagged instances and align them according to the positions of $Slot_0$. As a result, we obtain a virtual vector *Pa* representing the contextual soft pattern rule as:

$<Slot_{-k}, \dots , Slot_{-2}, Slot_{-1}, Slot_0, Slot_1, Slot_2, \dots, Slot_k: Pa>$ (2)

where $Slot_i$ is a vector of tokens occurring in that slot with their probabilities of occurrence:

$<(token_{i1}, weight_{i1}), (token_{i2}, weight_{i2}) \dots .(token_{im}, weight_{im}): Slot_i>$ (3)

Here, $token_{ij}$ denotes any word, punctuation, syntactic or semantic tag contained in $Slot_i$, and $weight_{ij}$ gives the proportion of occurrences of the j^{th} token to the i^{th} slot. Figure 3 shows the generated soft pattern rules for the examples given in Figure 2.

```
(1) Training instances:
Time : <stime> NP_Time </stime>
VPact_be at <stime> NP_Time </stime>
NP_Day , NP_Date <stime> NP_Time </stime> - NP_Time
VPact_be at <stime> NP_Time </stime> , NP_Day , NP_Date
Time : <stime> NP_Time </stime> - NP_Time
```

(2) Soft pattern rules based on the instances:

…… $<Slot_{-2}>$	$<Slot_{-1}>$	$<Slot_0>$	$<Slot_1>$ …...
Time 0.4	: 0.4	NP_Time 1	- 0.67
VPact_be 0.4	at 0.4		, 0.33
, 0.2	NP_Date 0.2		

Figure 3: An excerpt of soft pattern rules

What results from the generalization process is a virtual vector *Pa* representing the soft pattern rule. The soft pattern vector *Pa* is then used to compute the degree of match for the unseen instances. The unseen instances are first pre-processed with the identical procedures as outlined in Section 3.1. Using the same window size *k*, the token fragment *S* surrounding the potential slot is derived:

$<token_{-k}, \dots, token_{-2}, token_{-1}, Potential\_Slot, token_1, token_2, \dots, token_k: S>$ (4)

The degree of match for the unseen instance against the soft pattern rules is measured by the similarity between the vector *S* and the virtual soft pattern vector *Pa*. In particular, the match degree is the combination of the individual slot content similarities and the fidelity degree of slot sequences measured by a bi-gram model (Cui, *et al.*, 2004).

When applying the soft pattern rules to automatically tag training instances, for each potential slot, we assign a target tag whose soft pattern rule gives the highest score beyond a pre-defined threshold.

3.3 Hard Pattern Rule Induction

We employ a pattern rule induction algorithm called GRID (Xiao, *et al.*, 2003) to generalize the hard pattern rules over all instances hand-tagged by users and automatically annotated by soft pattern rules. GRID is a supervised covering algorithm. It uses chunks as contextual slots and considers a context size of k slots around the tagged item as definition in Equation (1).

Given the cluster of training instances for a specific slot type, GRID aligns all the instances according to the central slot ($Slot_0$) as is done in soft pattern rules. For each context slot, we store all possible representations of slot units as listed in Table 1 at the levels of lexical, syntactic and semantic simultaneously. Thus, we obtain a global context feature representation for the whole training corpus as shown in Figure 4. GRID records the occurrences of the common slot features at a specific position as e_{ij} ($i = -k, \ldots, -1, 0, 1, \ldots, k$; j^{th} feature for $Slot_i$).

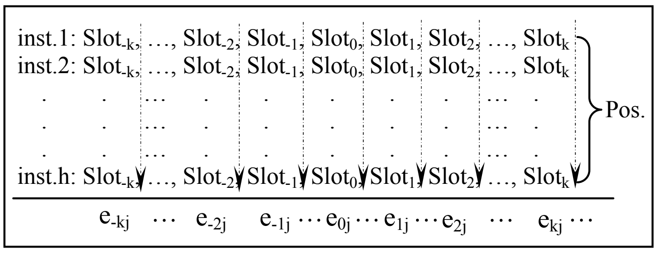
inst.1: Slot-k, …, Slot-2, Slot-1, Slot0, Slot1, Slot2, …, Slotk
inst.2: Slot-k, …, Slot-2, Slot-1, Slot0, Slot1, Slot2, …, Slotk
inst.h: Slot-k, …, Slot-2, Slot-1, Slot0, Slot1, Slot2, …, Slotk
Pos.
e-kj … e-2j e-1j … e0j … e1j … e2j … ekj …

Figure 4: Global distribution of positive instances

GRID generates a pattern rule $r_k(f)$ by adding slot features into the feature set f. The quality of $r_k(f)$ is determined not only by its coverage in the positive training set but also by the number of instances in the negative set that it covers which would be regarded as errors. We define the remaining instances which are not annotated by human and soft pattern rules as negative instances.

We use a modified *Laplacian* expected error (Soderland, 1999) to define the quality of the rule as follows:

$$Laplacian(r_k(f)) = \frac{n_k + 1}{n_k + p_{k1} + 0.7 \times p_{k2} + 1} \quad (5)$$

where p_{k1} denotes the number of instances covered by rule $r_k(f)$ in the manually annotated set, and p_{k2} denotes the number of instances covered by the rule $r_k(f)$ in the automatically annotated set. n_k is the number of negative examples or errors covered by the rule. We consider all the manually annotated instances as correctly tagged and thus we put more weight on them than on the automatically annotated data set.

Instead of generalizing a rule from a specific instance as is done in most existing pattern rule induction algorithms, GRID examines the global feature distribution on the whole set of training examples in order to make better decision on rule induction. Each time, GRID selects top w features (in terms of the e_{ij} values) and selects slot feature f_{ij} with the minimum *Laplacian* value of the rule ($r_k(f \cup f_{ij})$) according to Equation (5) to induce pattern rules (Xiao, *et al.,* 2003).

We use GRID to generate rules that cover all seed instances and discard some rules generated from the automatically tagged instances whose *Laplacian* value is greater than a preset threshold.

3.4 Cascading Matching of Hard and Soft Pattern Rules

After we have obtained the set of hard pattern rules and the set of soft pattern rules through the bootstrapping rule induction process, we apply both sets of rules in a cascaded way to assign appropriate tag to potential slots in new instances. The tag assigned to the given test instance t is selected by:

1) tag_g matched by GRID $rule_g$;
2) If not matched by any GRID rule,
   tag_i $\quad \arg\max_{Pa_i \in Pa} \Pr(t \mid Pa_i) > \theta$

We apply the high-precision hard pattern rules generated by GRID first. In this case, we assign tag_g to the instance if it matches $rule_g$. In order to increase the coverage of the hard pattern rules, we allow up to one shift in the context vectors of new test instances when matching the instances against the hard pattern rules.

For the remaining test instances that are not matched by any of the hard pattern rules, we score them using the soft pattern rules. A test instance is assigned tag_i if it has the highest conditional probability of having t given the soft pattern rule i (represented by vector Pa_i) which is greater than a pre-defined threshold θ among all the soft pattern rules.

4 Evaluation

To verify the generality and effectiveness of our bootstrapping framework, we have conducted two experiments on free and semi-structured texts. In our supervised IE system using GRID (Xiao, *et al.,* 2003), we had done some trial experiments to examine the effect of varying the different context length k, and found the IE performance became stable when the context length reached 4. As such, we set the context length k to 4 for all subsequent experiments.

4.1 Results on free text corpus

The first evaluation was conducted on the terrorism domain using the MUC-4 free text corpus (MUC-4, 1992). We employed the same evaluation measures as that in (Riloff, 1996; Xiao, *et al.,* 2003). The target extracted slots were "perpetrator" (Perp.), "victim" (Vic.) and "target" (Tar.). We varied the number of the human-annotated instances from the 772 relevant documents set (the standard training documents for MUC-4 plus TST1 and TST2) used in supervised IE learning. The manual annotation was guided by the associated answer keys given in the MUC-4 corpus. During testing, we used the 100 texts comprising 25 relevant and 25 irrelevant texts from the TST3 test set, and 25 relevant and 25 irrelevant texts from the TST4 test set.

Following the procedure discussed in Section 2, we repeated the automated annotation process several times ($i \geq 1$ in Figure 1). To examine the variation of performance along with the changing of the number of iterations, we plotted the average F_1 measures of the three target slots against the iteration number (see Figure 5). We also varied the number of manually tagged instances that were utilized as seed instances for starting the bootstrapping process. As can be seen in Figure 5, the results improved as the number of iterations increased. The system achieved a steady performance when the number of iterations reached four. Accordingly in the next experiments, we considered the system's performance based on four bootstrapping iterations.

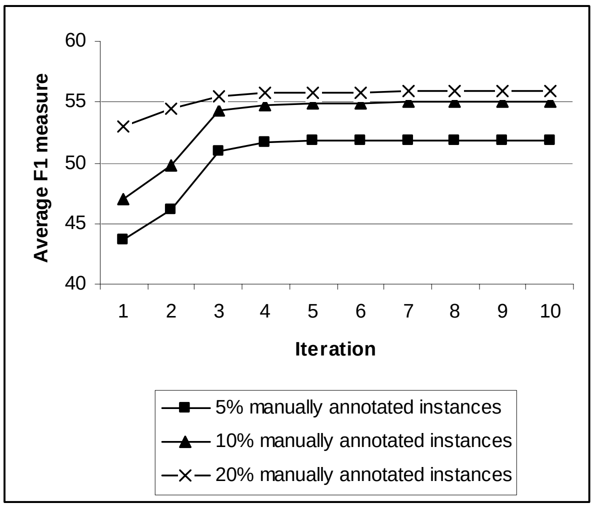

Figure 5: Effect of the number of iterations

Table 2 shows the performance of the system on the test data in terms of F_1-measure (with recall/precision value in the brackets) using various amounts of manually tagged data after four iterations. To demonstrate the effectiveness of the combination of hard and soft pattern rules, we also ran four iterations using only soft pattern rules (SP) and another four with only GRID rules.

From Table 2, we can draw the following conclusions:

(a) The cascaded learner by combining SP and GRID outperforms the learner SP or GRID alone. The soft pattern learner (SP) alone cannot achieve good precision while the hard pattern learner (GRID) alone cannot achieve high recall with a small set of hand-annotated instances.

	Perp.	Vic.	Tar.
5%(SP)	36 (42/32)	45 (49/42)	42 (47/38)
5%(GRID)	34 (35/33)	44 (40/49)	39 (36/43)
5%(SP+GRID)	47 (49/45)	58 (59/57)	50 (50/50)
10%(SP)	38 (45/33)	46 (51/42)	45 (49/42)
10%(GRID)	37 (39/35)	46 (41/52)	44 (41/47)
10%(SP+GRID)	50 (53/47)	61 (63/59)	53 (52/54)
20%(SP)	40 (46/35)	48 (54/43)	47 (50/44)
20%(GRID)	40 (41/39)	47 (44/50)	47 (45/49)
20%(SP+GRID)	51 (52/50)	62 (63/61)	54 (55/53)
AutoSlog-TS	38 (53/30)	48 (62/39)	47 (58/39)
supervised(GRID)	52 (48/57)	62 (58/67)	56 (51/62)

Results presented in terms of F_1(recall/precision).

Table 2: Results on free text domain

(b) Compared with another weakly supervised IE system in the same domain, AutoSlog-TS (Riloff, 1996), our cascaded learner outperforms it with the use of only 5% of the manually tagged instances.

(c) As the percentage of the hand-annotated instances increases from 5% to 20%, the performance of the cascaded learner (SP+GRID) increases steadily, indicating that the bootstrapping process is stable and consistent.

(d) With 20% of hand-tagged training instances, the performance of the cascaded learner approaches that of the fully supervised IE tagger. When more manually tagged instances (>20%) are used, the performance of the cascaded learner becomes steady.

(e) Looking at the instances automatically tagged by the soft pattern rules, we found that about 75% instances are correctly annotated in the first and second iteration. The percentage of correctly tagged instances by soft pattern rules increases to 90% when the bootstrapping process runs for four times. The percentage increase verifies that our automated annotation can provide relatively accurate training instances for later rule induction.

Nevertheless, our system missed some cases which needed deeper NLP analysis. For example, given a test sentence "THEY ARE THE TOP MILITARY AND POLITICAL FIGURES IN ALFREDO CRISTIANI'S ADMINISTRATION.", the system could not identify "ALFREDO CRISTIAN'S ADMINISTRATION" as the "perpetrator". If we could associate the previously found "perpetrator" (maybe located far away) to "they", then we might be able to infer that the "ALFREDO CRISTIAN'S ADMINISTRATION" is the "perpetrator" too.

4.2 Results on semi-structured corpus

The second experiment was conducted on semi-structured text documents. We used the CMU seminar announcements[2] for the evaluation. The IE task for this domain is to extract the entities of "speaker" (SP), "location" (LOC), "starting time" (ST), and "ending time" (ET) from a seminar announcement. There were 485 seminar announcements. In the supervised IE experiments, we made five runs and in each run we used one half for training and the other half for testing. Similarly, to evaluate our weakly supervised learning framework, we did five trials as well. In each run, we varied the percentage of manually annotated instances for training in the supervised experiments. Table 3 shows the performance (the average F_1 measure and recall/precision for five runs) of the system with different percentage of manually tagged instances used to start the training. We also compare the performances between the single learners and the cascaded learner. All results are based on four bootstrapping iterations.

	SP	LOC	ST	ET
5%(SP)	70 (74/66)	65 (70/61)	94 (95/93)	90 (93/88)
5%(GRID)	68 (65/72)	61 (59/64)	93 (91/94)	89 (86/92)
5%(SP+GRID)	82 (83/81)	73 (74/72)	98 (98/98)	94 (96/92)
10%(SP)	72 (75/70)	68 (72/64)	96 (96/95)	93 (94/92)
10%(GRID)	72 (67/77)	67 (63/72)	95 (94/96)	93 (91/96)
10%(SP+GRID)	84 (84/83)	75 (75/74)	99 (99/99)	95 (97/94)
20%(SP)	75 (77/74)	71 (75/67)	97 (97/97)	95 (96/95)
20%(GRID)	75 (69/82)	71 (66/77)	97 (95/99)	95 (94/96)
20%(SP+GRID)	85 (85/85)	76 (76/75)	99 (99/99)	96 (97/95)
supervised (GRID)	86 (84/88)	76 (73/80)	99 (99/100)	96 (95/97)

Results presented in terms of F_1(recall/precision).

Table 3: Results on semi-structured data

From Table 3, we make the following observations:

(a) The cascaded learner with two pattern learners significantly outperforms the learner SP or GRID alone as in the case of free text corpus. With 10% of hand-tagged instances, the cascaded learner (SP+GRID) approaches the performance of the fully supervised IE tagger. Also the performance of the cascaded learner increases steadily when the number of hand-tagged instances increases from 5% to 20%.

(b) With more hand-annotated instances (>20%), the performance of the bootstrapping system with the cascading use of SP and GRID becomes stable and consistent.

(c) Soft pattern rules tag 90% of the instances correctly, as we found out in our random checks.

The lower performance of our system on the "location" slot is mainly due to the use of a general named entity recognizer which is good at identifying common locations such as cities, mountains etc. In seminar announcements, many locations are room numbers such as "WeH 8220"; thus, we missed out some seminar venues.

5 Related Work

Many hard pattern rule inductive learning systems have been developed for information extraction from free texts or semi-structured texts. Specifically, AutoSlog-TS (Riloff, 1996) generates extraction patterns using annotated text and a set of heuristic rules and it eliminates the dependency on tagged text and only requires the pre-classified texts as input. WHISK (Soderland, 1999) induces multi-slot rules from a training corpus top-down. It is designed to handle text styles ranging from highly structured text to free text. WHISK performs rule induction starting from a randomly selected seed instance. $(LP)^2$ (Ciravegna, 2001) is a covering algorithm for adaptive IE systems that induces symbolic rules. In $(LP)^2$, training is performed in two steps: first, a set of tagging rules is learned to identify the boundaries of slots; next, additional rules are induced to correct mistakes in the first step of tagging. In contrast to their work, GRID utilizes global feature distribution to induce pattern rules and uses chunk as the context unit.

Nahm and Mooney (2001) proposed the learning of soft matching rules from texts by combining rule-based and instance-based learning. Words in each slot are generalized by traditional rule induction techniques and test instances are matched to the rules by their cosine similarities. The learning of soft pattern rules in this paper augments the soft matching method advocated by Nahm and Mooney (2001) by combining lexical tokens alongside syntactic and semantic features and adopting a probabilistic framework that combines slot content and sequential fidelity in computing the degree of pattern match.

The bootstrapping scheme using the co-training (Blum and Mitchell, 1998) technique has been widely explored for IE tasks in recent years. Collins and Singer (1999) presented several techniques using co-training schemes for Named Entity (NE) extraction seeded by a small set of manually crafted NE rules. Riloff and Jones (1999) presented a multi-level bootstrapping algorithm that generates both the semantic lexicon and extraction patterns simultaneously. Yangarber (2003) proposed a counter-training approach to provide natural stopping criteria for unsupervised

[2] http://www.isi.edu/info-agents/RISE/repository.html

learning.

Our framework of combining two pattern learners is close to Niu, *et al.* (2003) in which two successive learners are used to learn named entities classifiers starting from a small set of concept-based seed words. The bootstrapping procedure is implemented as training a decision list and an HMM classifier sequentially. The HMM classifier uses the training corpus automatically, tagged by the first learner, i.e., the decision list learner. Our work differs from Niu, *et al.* (2003) in two ways. First, we repeat the automatic annotation process until it satisfies the stopping criteria. Second, we apply different patterns (hard and soft pattern rules) in both the training and test phases.

6 Conclusion

We have presented a novel bootstrapping approach for information extraction by the cascading use of soft and hard pattern rules. Our framework takes advantages of the high-recall of soft pattern rules and the high-precision of hard pattern rules. We use soft pattern rules to automatically annotate more training instances so as to provide a more comprehensive basis for hard pattern rule induction. The integration of soft pattern matching in the extraction phase also provides more target entities from test instances that would otherwise be missed by hard pattern matching. With much less manual input, the proposed bootstrapping system approaches the performance obtained by fully supervised learning on both semi-structured and free texts corpora.

7 Acknowledgement

The authors would like to thank Alexia Leong for proofreading this paper. The third author is supported by Singapore Millennium Foundation Scholarship (ref no. 2003-SMS-0230).

References

A. Blum and T. Mitchell. 1998. Combining Labeled and Unlabeled Data with Co-training. *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT-98),* pages 92-100.

T.-S. Chua and J. Liu. 2002. Learning Pattern Rules for Chinese Named Entity Extraction. *Proceedings of the 18th National Conference on Artificial Intelligence. (AAAI-02)*, pages 411-418.

F. Ciravegna. 2001. Adaptive Information Extraction from Text by Rule Induction and Generalisation. *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-2001)*, pages 1251-1256.

M. Collins and Y. Singer. 1999. Unsupervised Models for Named Entity Classification. *Proceedings of the 1999 Joint SIGDAT Conference on EMNLP and VLC.*

H. Cui, M.-Y. Kan and T.-S. Chua. 2004. Unsupervised Learning of Soft Patterns for Definitional Question Answering. *Proceedings of 13th World Wide Web Conference. (WWW-04),* pages 90-99.

MUC-4, 1992. *Proceedings of the Fourth Message Understanding Conference*. San Mateo, CA: Morgan Kaufmann. 1992.

I. Muslea. 1999. Extraction Patterns for Information Extraction Tasks: A Survey. *The AAAI-99 Workshop on Machine Learning for Information Extraction.*

U. Y. Nahm and R. J. Mooney. 2001. Mining Soft Matching Rules from Textual Data. *Proceedings of the 17th International Joint Conference on Artificial Intelligence. (IJCAI-01),* pages 979-986.

C. Niu, W. Li, J. Ding and R. K. Srihari. 2003. A Bootstrapping Approach to Named Entity Classification Using Successive Learners. *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics. (ACL-03)*, pages 335-342.

E. Riloff. 1996. Automatically Generating Extraction Patterns from Untagged Text. *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI-96)*, pages 1044-1049.

E. Riloff and R. Jones, 1999, Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping, *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, pages 474-479.

S. Soderland. 1999. Learning Information Extraction Rules for Semi-structured and Free Text. *Machine Learning*, vol.34, pages 233-272.

J. Xiao, T.-S. Chua and J. Liu. 2003. A Global Rule Induction Approach to Information Extraction. *Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence. (ICTAI-03)*, pages 530-536.

R. Yangarber. 2003. Counter-Training in Discovery of Semantic Patterns. *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-03)*, pages 343-350.