Applying Type-Oriented ILP to IE Rule Generation

Yutaka Sasaki
NTT Communication Science Laboratories
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0237 Japan
Phone: +81 774-93-5360, FAX: +81 774-93-5385
Email: sasaki@cslab.kecl.ntt.co.jp

Abstract

This paper describes our approach to applying type-oriented *inductive logic programming (ILP)* to *information extraction (IE)* tasks and the latest experimental results in learning IE rules from the data generated from 100 newspaper articles. Information extraction involves extracting key information from text corpus in order to fill empty slots of given *templates*. A bottle neck in building IE systems is that constructing and verifying IE rules is labor intensive and time consuming. To automatically generate IE rules, we employ an ILP system RHB$^+$ that learns logic programs whose variables have type information. After giving our approach to applying ILP to information extraction tasks in detail, the process of learning IE rules is illustrated. Experiments were conducted on the data generated from 100 newspaper articles relating to release of new products. The results show high accuracy and precision of the learned rules. This indicate that type-oriented ILP has a high potential for the use of automatically generating IE rules.

Introduction

Information extraction (IE) techniques have been studied by many researchers and institutions in a series of Message Understanding Conferences (MUC). These conferences have addressed not only technical points but also have served as a forum for IE system contests, conducted by sharing the same articles and *templates*.

The IE tasks in this paper involved the MUC-3 style IE. The input for the information extraction task is an empty template and a set of natural language texts that describe a certain topic, such as corporate mergers or terrorist attacks in South America. Templates have a record-like data structure with slots which have slot names, *e.g.*, "company name" and "merger date", and slot values. The output is a set of filled templates. IE tasks are highly domain dependent so rules and dictionaries for filling values in the template slots depend on the domain.

One problem for IE system developers was that the systems depended on hand-made rules, which were not able to be easily constructed and changed. For example, Umass/MUC-3 needed about 1500 person-hours of highly skilled labor to build the IE rules and represent them as a dictionary (Lehnert *et al.*, 1992). New rules must be constructed from scratch when the target domain is changed.

To cope with this problem, some frontiers have studied methods for learning information extraction rules for years. Along this line, our approach is to apply an ILP system to the learning of IE rules by which information is extracted from case frames of news articles. The ILP system that we employed is the type-oriented *inductive logic programming* (ILP) system RHB$^+$(Sasaki and Haruno, 1997).

In this paper, we describe the results of applying the ILP system RHB$^+$ to the IE in the "new product release"domain in the following manner. Section 2 describes our approach to IE tasks. Section 3 shows experimental results in the new product release domain. Section 4 describes related work and Section 5 concludes this paper.

Our approach to IE tasks

This section describes our approach to IE tasks in the new product release domain. Figure 1 is a block diagram of our IE system using a type-oriented ILP system RHB$^+$ (Sasaki and Haruno, 1997). First, training articles are analyzed and converted into *case frames*, which are represented as atomic formulae. Training templates are prepared by hand as well. The ILP system learns IE rules in the form of logic programs with type information from the case frames and filled templates. To extract key information from a new article, case frames that are generated from the article are matched by the IE rules. Information extracted is filled into the template slots, such as "company name" and "product name".

Overview of employed learner RHB$^+$

Based on (Sasaki and Haruno, 1997), this section summarizes employed type-oriented ILP system RHB$^+$.

The input of RHB$^+$ is a set of positive examples and background knowledge including type hierarchy (or taxonomy). The type hierarchy is represented as a tree structure in which each node is a type. The

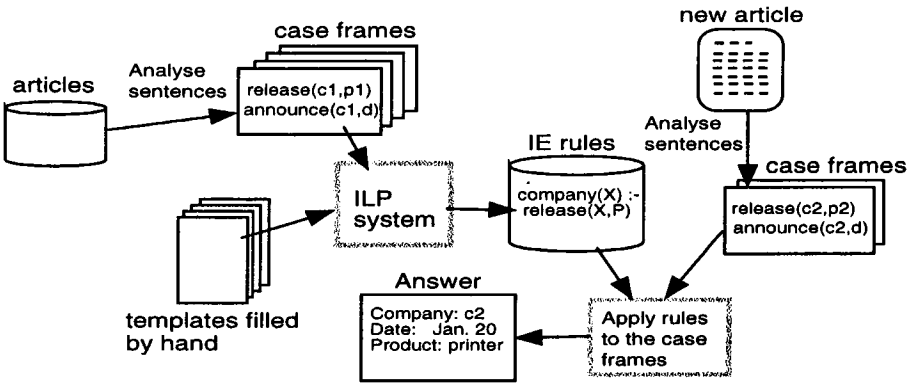

Figure 1: Block diagram of IE using ILP

direct links between the nodes represent *is_a* relations and the root of the tree, described as $\top$, is the most general type. The output is a set of clauses having variable with type information.

RHB$^+$ has two features indispensable to the learner for IE tasks: (1) it learns logic programs based on τ-terms; (2) learns from only positive examples. τ-terms are the restricted form of ψ-terms used in LOGIN (Aït-Kaci and Nasr, 1986) and LIFE (Aït-Kaci *et al.*, 1994). Informally, τ-terms are Prolog terms whose variables are replaced with variable *Var* of type T, which is denoted as *Var:T*. For example, a learned hypothesis can be

speak(X:human, Y:language) :–
grew_in(X,Z:country), official_lang(Z, Y).

Moreover, a function is allowed to have labels or keywords for readability. A combination of bottom-up and top-down approaches is employed, following the result described in (Zelle *et al.*, 1994).

τ-term

Formally, terms of a Horn clause (Lloyd, 1987) were extended to terms with types, called τ-terms.

Definition 1 (τ-*terms*)

1. *An individual constant is a τ-term.*
2. *A type is a τ-term.*
3. *If τ is a type or constant and X is a variable, $X{:}\tau$ is a τ-term.*
4. *If $t_1, ..., t_n$ are τ-terms, $l_1, ..., l_n$ are labels, and f is a function symbol of arity n, then $f(l_1 \Rightarrow t_1, ..., l_n \Rightarrow t_n)$ is a τ-term.*

Lgg with types

In the definition of the *least general generalization (lgg)* (Plotkin, 1969), the definition of the term *lgg* should be extended to *lgg* of τ-terms. Other definitions of *lgg* are original. Note that $lub(t, s)$ is a function which returns $U{:}u$ where u is the *least upper bound* of t and s according to the type hierarchy if t and s are types, otherwise it returns $V{:}\top$ with a new variable V. For a concice definition, we use the operator prime (') which is defined as $(X{:}s)'{=}s$, and $t' = t$ if t is a constant, type or function with no variable attached.

Definition 2 lgg of terms with types

1. $lgg(t, t) = t$.
2. $lgg(f(l_1 \Rightarrow s_1, .., l_n \Rightarrow s_n), f(l_1 \Rightarrow t_1, .., l_n \Rightarrow t_n) = f(l_1 \Rightarrow lgg(s_1, t_1), ..., l_n \Rightarrow lgg(s_n, t_n))$.
3. *$lgg(s, t) = u$ for term s and t of different symbols or with different function names, where the tuple (s, t, u) is in the history H.*
4. *$lgg(s, t) = u$ for term s and t of different symbols or with different function names, where $u = lub(s', t')$ and (s, t, u) is not in H. Then add (s, t, u) into H.*

Algorithm

The outer loop of RHB^+ finds covers of the given positive examples P in a greedy manner. It constructs clauses one by one by calling $inner\_loop(P, P_0, BK)$ which returns a hypothesis clause, where P_0 is original positive examples and BK is background knowledge. Covered examples are removed from P in each cycle; P_0 remains unchanged. Algorithm 1 shows the algorithm of the inner loop.

The special feature of RHB$^+$ is the *dynamic type restriction by positive examples* during clause construction. The restriction uses positive examples currently covered in order to determine appropriate types. For each variable X_i appearing in the clause, RHB^+ computes the *lub* of all types bound to X_i when covered positive examples are unified with the current head in turn.

The search heuristic PWI is weighted informativity employing the Laplace estimate (Cestnik, 1990). Let $T = \{Head :{-} Body\} \cup BK$.

$$PWI(T) = -\frac{1}{|\hat{P}|} \times \log_2 \frac{|\hat{P}|x + 1}{|Q(T)| + 2},$$

Table 1: Sample sentences

Article id	Sentence
#1	"ABC Corp. this week announced that it will release a color printer on Jan. 20."
#2	"XYZ Corp. released a color scanner last month."

Algorithm 1 *inner_loop*(P, P_0, BK)

1. *Given positive examples P,P_0, background knowledge BK.*
2. *Decide types of variables in a head by computing the typed least general generalizations (lgg) of N pairs of elements in P, and select the most general head as Head.*
3. *If the stopping condition is satisfied, return Head.*
4. *Let Body be empty.*
5. *Create a set of all possible literals L using variables in Head and Body.*
6. *Let $BEAM$ be top K literals l_k of L with respect to positive weighted informativity PWI.*
7. *Do later steps, assuming that l_k is added to Body for each literal l_k in $BEAM$.*
8. *Dynamically restrict types in Body by calling the dynamic type restriction positive examples.*
9. *If the stopping condition is satisfied, return (Head :- Body).*
10. *Goto 5.*

where $|\hat{P}|$ denotes the number of positive examples covered by T and $Q(T)$ is the empirical content.

Type information is used for computing $Q(T)$. Let Hs be a set of instances of $Head$ generated by proving $Body$ using backtracking. $|\tau|$ is defined as the number of leaves under type τ in the type hierarchy. When τ is a constant, $|\tau|$ is defined as 1.

$$|Q(T)| = \sum_{h \in Hs} \prod_{\tau \in Types(h)} |\tau|,$$

where $Types(h)$ returns the set of types in h.

The stopping condition also utilizes $|Q(T)|$ in the computation of the *Model Covering Ratio (MCR)*:

$$MCR(T) = \frac{|\hat{P}|}{|Q(T)|}.$$

Illustration of a learning process

Let us look at the process of learning IE rules in more detail. Key information in this domain is as follows.

- name of company that released the product
- product name
- release date
- announce date
- product's price

The following table shows a sample template for articles reporting a new product release.

Slot name	Value
company:	ABC Corp.
product:	color printer
release date:	Jan. 20
announce date:	last week
price:	$530

Now, we examine the two short notices of release of new products in Table 1. The template for these two articles is as follows.

template
1. article id:
2. company:
3. product:
4. release date:

Suppose that the following case frames are obtained from Article 1.

```
(c1) announce( article => 1,
       tense => past,
       time  => "this week",
       agent => "ABC Corp.",
      object => (c2) ).

(c2)  release( article => 1,
       tense => future,
       time  => "Jan. 20",
       agent => "ABC Corp.",
      object => "color printer" ).
```

The filled template for Article 1 is as follows.

Template 1
1. article id: 1
2. company: ABC Corp.
3. product: printer
4. release date: "Jan. 20"

Suppose that the following case frames are obtained from Article 2.

```
(c3) release( article => 2,
       tense => past,
       time  => "last month",
       agent => "XYZ Corp.",
      object => scanner ).
```

The filled template for Article 2 is as follows.

Template 2
1. article id: 2
2. company: XYZ Corp.
3. product: scanner
4. release date: last month

The rules for extracting "company name", "product name" and "release date" are listed below.

```
company(article-number => Art:number,
     name => Co: organization    )

  :- release( article => Art,
         tense => tense,
         time  => D,
         agent => Co,
        object => P     ).

product( article-number => Art:number,
          name => P: machine        )

  :- release( article => Art,
         tense => tense,
         time  => D,
         agent => Co,
        object => P     ).

date( article-number => Art:number,
     release-date => D: date        )

  :- release( article => Art,
         tense => tense,
         time  => D,
         agent => Co,
        object => P     ).
```

Now, we have the following case frame extracted from new Article 3: "JPN Corp. has released a new CD player."[1]

```
(c4) release( article => 3,
       tense => perfect_present,
       time  => nil,
       agent => "JPN Corp.",
      object => "CD player" ).
```

Applying the three IE rules, we can obtain the filled template for Article 3.

Template 3
1. article id: 3
2. company: JPN Corp.
3. product: CD player
4. release date:

Experimental results

Natural language processing tools

We used the commercial-quality morphological analyzer, parser and semantic analyzer actually used in the Japanese-English machine translation system ALT-J/E (Ikehara *et al.*, 1993). We also used the type hierarchy hand-crafted for ALT-J/E. The hierarchy is a sort of concept thesaurus represented as a tree structure in which each node is called a category (*i.e.*, a type). An edge in the tree represents an *is_a* relation among the categories. The current version of the type hierarchy is 12 levels deep and contains about 3000 category nodes.

The semantic analysis provides parse trees with case and semantic tags. We developed a logical form translator that generates case frames expressed as atomic formulae from the parse trees.

Setting of experiments

We extracted articles related to the release of new products from a one-year newspaper corpus written in Japanese [2]. One-hundred articles were randomly selected from 362 relevant articles. The template we used consisted of five slots: company name, product name, release date, announce date, and price. We also filled one template for each article. Only the sentences including words in the slots of the filled templates were chosen (which eliminates irrelevant sentences and analyzed. After that, tagged parse trees were converted into atomic formulae representing case frames. The case frames were given to the learner as background knowledge and the filled templates were given as positive examples. Precision and recall, the standard metrics for IE tasks, are counted by using the remove-one-out cross validation on the 100 examples for each item.

Results

Table 2 shows the results of our experiment. Overall, accuracy was very high. 77-88% recall was achieved with all case frames including errors in case selection and semantic tag selection. With only correct case frames, 88-94% recall was achieved.

It is important that the extraction of five different pieces of information showed good results. This indicates that the ILP system RHB$^+$ has a high potential in IE tasks.

Related work

Previous researches on generating IE rules from texts with templates include AutoSlog-TS (Riloff,1996), CRYSTAL (Soderland *et al.*, 1995), PALKA (Kim *et al.*, 1995), LIEP (Huffman, 1996), RAPIER (Califf and Mooney, 1997). In our approach, we use type-oriented ILP system RHB$^+$ independent of natural language analysis. This point defferenciates our approach from the others. That is, learned logic programs may have several atomic formulae in the bodies. Our approach is different from simple generalization of a single case frame. That is, information in some case frames could be complement to archive an appropriate generality of an IE rule.

Sasaki (Sasaki, 1998) reported results of preliminary expreiments on learning IE rules to extract information from only twenty articles using the preliminary version

[1] We always assume *nil* for the case that is not included in the sentence.

[2] We used the Mainichi Newspapers articles of 1994 with permission.

Table 2: Learning results of new product release

	company	product	release date	announce date	price
Precision (all case frames)	82.8%	90.9%	98.9%	100.0%	91.3%
Precision (correct case frames)	90.0%	100.0%	98.9%	100.0%	92.6%
Recall (all case frames)	80.9%	70.0%	84.2%	88.2%	76.8%
Recall (correct case frames)	90.0%	88.6.0%	93.8%	88.2%	87.5%
Average time (sec.)	302.0	396.4	466.9	46.1	202.4

ILP of the system, which was capable of most features of ψ-terms. The hypothesis language of the preliminary version of the learner has more representation power than that of RHB$^+$ but this prower comes at the expense of learning speed. When almost all case frames of the articles are within the range of RHB$^+$'s hypothesis language, it is a better to use RHB$^+$. Sasaki (Sasaki and Haruno, 1997) also applied RHB$^+$ to the extraction of the number of deaths and injuries from twenty five articles relating to accidents. That experiment was enough to assess the performance of the learner, but not sufficient to evaluate its feasibility in IE tasks.

Conclusions and remarks

This papar decsribed an application of the type-oriented ILP system RHB$^+$ to the generation of information extraction rules. Experimets were concucted on the data generated from 100 news articles in the domain of new product release. The results showed that very high precision, 77-88% recall with all case frames, and 88-94% recall with correct case frames. It is worth remarking that the extraction of five different pieces of information showed good results. This indicates that our learner RHB$^+$ has a high potential in IE tasks.

References

H. Aït-Kaci and R. Nasr, LOGIN: A logic programming language with built-in inheritance, *J. Logic Programming*, 3, BP.185-215, 1986.

H. Aït-Kaci, B. Dumant, R. Meyer, A. Podelski, and P. Van Roy, *The Wild Life Handbook*, 1994.

B. Cestnik, Estimating probabilities: A crucial task in machine learning, *ECAI-90*, pp.147-149, 1990.

S. Ikehara, M. Miyazaki, and A. Yokoo, Classification of language knowledge for meaning analysis in machine translations, *Transactions of Information Processing Society of Japan*, vol. 34, pp.1692-1704, 1993 (in Japanese).

J.-T. Kim and D. I. Moldovan, Acquisition of Linguistic Patterns for Knowledge-Based Information Extraction, IEEE TKDE, Vol.7, No.5, pp. 713-724, 1995.

W. Lehnert, C. Cardie,D. Fisher, J. McCarthy, E. Riloff and S. Soderland, University of Massachusetts: MUC-4 Test Results and Analysis, in Proc. of MUC-4, pp.151-158, 1992.

J. Lloyd, *Foundations of Logic Programming*, Springer, 1987.

M. E. Califf and R. J. Mooney, Relational Learning of Pattern-Match Rules for Information Extraction, ACL-97 Workshop in Natural Language Learning, 1997.

G. Plotkin, A note on inductive generalization, in B. Jeltzer *et al.* eds., Machine Intelligence 5, pp.153-163, Edinburgh University Press, 1969.

E. Riloff, Automatically Generating Extraction Pattern from Untagged Text, AAAI-96, pp. 1044-1049, 1996.

Y. Sasaki and M. Haruno, RHB$^+$: A Type-Oriented ILP System Learning from Positive Data, IJCAI-97, pp.894-899, 1997.

Y. Sasaki, Learning of Information Extraction Rules using ILP — Preliminary Report, *The Second International Conference on The Practical Application of Knowledge Discovery and Data Mining*, London, 1998.

S. B. Huffman, Learning Information Extraction Patterns from Examples, Statistical and Symbolic Approaches to Learning for Natural Language Processing, pp. 246-260, 1996.

S. Soderland, D. Fisher, J. Aseltine, W. Lenert, CRYSTAL: Inducing a Conceptual Dictionary, IJCAI-95, pp.1314-1319, 1995.

J. M. Zelle and R. J. Mooney, J. B. Konvisser, Combining Top-down and Bottom-up Methods in Inductive Logic Programming, *ML-94*, pp.343-351, 1994.