Appears in *Working Notes of the IJCAI-97 Workshop on Frontiers of Inductive Logic Programming*, pp.7-11, Nagoya, Japan, August 1997

Applying ILP-based Techniques to Natural Language Information Extraction: An Experiment in Relational Learning

Mary Elaine Califf and Raymond J. Mooney
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712
{mecaliff,mooney}@cs.utexas.edu

1 Introduction

In complex and context-rich domains, inductive logic programming (ILP) has some advantages over propositional, or feature-based, machine learning algorithms. The feature-based systems require that the examples be reduced to a finite, manageable set of features. Development of such a set of features can require significant representation engineering and may still exclude important contextual information. A first order logic representation can represent a richer set of features and more easily capture contextual information. ILP also allows the use of background knowledge, and the resulting rules are often more comprehensible. The comprehensibility of symbolic rules makes it easier for the system developer to understand and verify the resulting system and perhaps even edit the learned knowledge [Cohen, 1996].

One domain with the complexity to make relational learning preferable to feature-based learning is natural language processing (NLP). Detailed experimental comparisons of ILP and feature-based induction have demonstrated the advantages of relational representations in two language related tasks, text categorization [Cohen, 1995] and generating the past tense of an English verb [Mooney and Califf, 1995].

However, for some NLP tasks, first order logic representations may be very difficult to produce. One such task is information extraction (IE), in which specific pieces of information are extracted from natural language documents. To actually use inductive logic programming to learn rules for this task, one would have to be able to robustly produce a first order representation of the original documents. However, relational learning does not have to limited to first order logic representations [Blockeel and deRaedt, 1996]. Therefore, we have chosen instead to apply ILP-based techniques to a rule representation more suited to the task. Using only a corpus of documents paired with filled templates, RAPIER (Robust Automated Production of Information Extraction Rules) learns unbounded Eliza-like patterns [Weizenbaum, 1966] that utilize limited syntactic information, such as the output of a part-of-speech tagger. Induced patterns can also easily incorporate semantic class information, such as that provided by WordNet [Miller *et al.*, 1993].

The remainder of this paper is organized as follows. Section 2 presents background material on IE and describes the specific ILP systems which inspired our algorithm. Section 3 describes RAPIER's rule representation and learning algorithm. Section 4 presents and analyzes results obtained on extracting information from messages posted to the newsgroup `misc.jobs.offered`. Section 5 presents our conclusions.

2 Background

2.1 Information Extraction

Information extraction is the task of locating specific pieces of data from a natural language document, and has been the focus of ARPA's Message Understanding Conferences (MUC) [Lehnert and Sundheim, 1991; ARPA, 1992; 1993]. Usually the data to be extracted is described by a template specifying a list of slots to be filled. For example, Figure 1 shows part of a job posting, and the corresponding slots of the filled computer-science job template.

IE can be useful in a variety of domains. The various MUC's have focused on domains such as Latin American terrorism, joint ventures, microelectronics, and company management changes. Others have used IE to track medical patient records [Soderland *et al.*, 1995] or company mergers [Huffmann, 1996]. The general task considered in this paper is extracting information from postings to USENET newsgroups, such as job announcements.

Posting from Newsgroup

```
Telecommunications. SOLARIS Systems
Administrator. 38-44K. Immediate need

Leading telecommunications firm in need
of an energetic individual to fill the
following position in the Atlanta
office:

  SOLARIS SYSTEMS ADMINISTRATOR
  Salary: 38-44K with full benefits
  Location: Atlanta Georgia, no
relocation assistance provided
```

Filled Template

```
computer_science_job
title: SOLARIS Systems Administrator
salary: 38-44K
state: Georgia
city: Atlanta
platform: SOLARIS
area: telecommunications
```

Figure 1: Sample Message and Filled Template

2.2 Related ILP Systems

The RAPIER algorithm employs a primarily bottom-up search and was inspired by three different ILP systems. Each of these are briefly described in order more clearly show how we use the learning techniques developed in ILP with an alternate representation.

GOLEM [Muggleton and Feng, 1992] employs a bottom-up algorithm based on the construction of relative least-general generalizations, *rlggs* [Plotkin, 1970]. The algorithm operates by randomly selecting pairs of positive examples, computing the determinate *rlggs* of each pair, and selecting the resulting consistent clauses with the greatest coverage of positive examples. That clause is further generalized by computing the *rlggs* of the clause with new randomly selected positive examples, and generalization terminates when the coverage of the best consistent clause stops improving.

The CHILLIN [Zelle and Mooney, 1994] system combines top-down (general to specific) and bottom-up ILP techniques. The algorithm starts with a most specific definition (the set of positive examples) and introduces generalizations which make the definition more compact. Generalizations are created by selecting pairs of clauses in the definition and computing LGGs. If the resulting clause covers negative examples, it is specialized by adding antecedent literals in a top-down fashion. The search for new literals is carried out in a hill-climbing fashion, using an information gain metric for evaluating literals. This is similar to the search employed by FOIL [Quinlan, 1990]. In cases where a correct clause cannot be learned with the existing background relations, CHILLIN attempts to construct new predicates which will distinguish the covered negative examples from the covered positives. At each step, a number of possible generalizations are considered; the one producing the greatest compaction of the theory is implemented, and the process repeats. CHILLIN uses the notion of *empirical subsumption*, which means that as new, more general clauses are added, all of the clauses which are not needed to prove positive examples are removed from the definition.

PROGOL [Muggleton, 1995] also combines bottom-up and top-down search. Using mode declarations provided for both the background predicates and the predicate being learned, it constructs a most specific clause for a random seed example. The mode declarations specify for each argument of each predicate both the argument's type and whether it should be a constant, a variable bound before the predicate is called, or a variable bound by the predicate. Given this most specific clause, PROGOL employs an A*-like search through the set of clauses containing up to k literals from that clause in order to find the simplest consistent generalization to add to the definition. Advantages of PROGOL are that the constraints on the search make it fairly efficient, especially on some types of tasks for which top-down approaches are particularly inefficient, and that its search is guaranteed to find the simplest consistent generalization if such a clause exists with no more than k literals. The primary problems with the system are its need for the mode declarations and the fact that too small a k may prevent PROGOL from learning correct clauses while too large a k may allow the search to explode.

3 RAPIER System

3.1 Rule Representation

RAPIER's rule representation uses patterns that make use of limited syntactic and semantic information, using freely available, robust knowledge sources such as a part-of-speech tagger and a lexicon with semantic classes, such as the hypernym links in WordNet [Miller *et al.*, 1993]. The initial implementation does not use a parser, primarily because of the difficulty of producing a robust parser for unrestricted text and because simpler patterns of the type we propose can represent useful extraction rules for at least some domains. The extraction rules are indexed by

Pre-filler Pattern:	Filler Pattern:	Post-filler Pattern:
1) word: leading	1) list: len: 2 tags: [nn, nns]	1) word: [firm, company]

Figure 2: A Rule Extracting an Area Filler from the Example Document

template name and slot name and consist of three parts: 1) a pre-filler pattern that must match the text immediately preceding the filler, 2) a pattern that must match the actual slot filler, and 3) a post-filler pattern that must match the text immediately following the filler. Each pattern is a sequence (possibly of length zero in the case of pre- and post-filler patterns) of *pattern items* or *pattern lists*. A pattern item matches exactly one word or symbol from the document that meets the item's constraints. A pattern list specifies a maximum length N and matches 0 to N words or symbols from the document that each must match the list's constraints. Possible constraints are: a list of words, one of which must match the document item; a list of part-of-speech (POS) tags, one of which must match the document item's POS tag; a list of semantic classes, one of which must be a class that the document item belongs to. Figure 2 shows a rule created by hand that extracts the `area` filler from the example document in figure reftemplate. This rule assumes that the document has been tagged with the POS tagger of [Brill, 1994].

3.2 The Learning Algorithm

As noted above, Rapier's algorithm primarily consists of a specific to general (bottom-up) search. First, for each slot, most-specific patterns are created for each example, specifying word and tag for the filler and its complete context. Thus, the pre-filler pattern contains an item for each word from the beginning of the document to the word immediately preceding the filler with constraints on the item consisting of the word and its assigned POS tag. Likewise, the filler pattern has one item from each word in the filler, and the post-filler pattern has one item for each word from the end of the filler to the end of the document.

Given this maximally specific rule-base, Rapier attempts to compress and generalize the rules for each slot. New rules are created by selecting two existing rules and creating a generalization. The aim is to make small generalization steps, covering more positive examples without generating suprious fillers, so a standard approach would be to generate the least general generalization (LGG) of the pair of rules. However, in this particular representation which allows for unconstrained disjunction, the LGG may be overly specific. Therefore, in cases where the LGG of two constraints is their disjunction, we want to create two generalizations: one would be the disjunction and the other the removal of the constraint. Thus, we often want to consider multiple generalization of a pair of items. This, combined with the fact that patterns are of varying length, making the number of possible generalizations of two long patterns extremely large, makes the computational cost of producing all interesting generalizations of two complete rules prohibitive. But, while we do not want to arbitrarily limit the length of a pre-filler or post-filler pattern, it is likely that the important parts of the pattern will be close to the filler. Therefore, we start by computing the generalizations of the filler patterns of the two rules and create rules from those generalizations. We maintain a list of the best n rules created and specialize the rules under consideration by adding pieces of the generalizations of the pre- and post-filler patterns of the two seed rules, working outward from the fillers. The rules are ordered using an information value metric [Quinlan, 1990] weighted by the size of the rule (preferring smaller rules). When the best rule under consideration produces no negative examples, specialization ceases; that rule is added to the rule base, and all rules empirically subsumed by it are removed. Specialization will be abandoned if the value of the best rule does not improve across k specialization iterations. Compression of the rule base for each slot is abandoned when the number of successive iterations of the compression algorithm which fail to produce a compressing rule exceed either a pre-defined limit or the number of rules for that slot. An outline of the algorithm appears in Figure 3 where *RuleList* is a prioritized list of no more than *Beam-Width* rules. The search is somewhat similar to a beam search in that a limited number of rules is kept for consideration, but all rules in *RuleList* are expanded at each iteration, rather than only the best.

4 Evaluation

This section presents preliminary results obtained with the current version of Rapier on computer-related job posting domain. The template contains 17 slots, including information about the employer, the location, the salary, and job requirements. Several of the slots, such as the languages and platforms used, can take multiple values. The current results do not employ semantic categories, only words and the results of Brill's POS tagger.

For each slot, S in the template being learned
    SlotRules = most specific rules from documents for S
    while compression has failed fewer than *lim* times
        randomly select 2 rules, *R1* and *R2*, from S
        find the set L of generalizations of the fillers of *R1* and *R2*
        create rules from L, evaluate, and initialize *RuleList*
        while best rule in *RuleList* produces spurious fillers and the weighted information value of the best rule is improving
            specialize each rule in *RuleList* with generalizations of the last n items of the pre-filler patterns of *R1* and *R2* and add specializations to *RuleList*
            specialize each rule in *RuleList* with generalizations of the first n items of the post-filler patterns of *R1* and *R2* and add specializations to *RuleList*
        if best rule in *RuleList* produces only valid fillers
            Add it to *SlotRules* and remove empirically subsumed rules

Figure 3: RAPIER Algorithm for Inducing IE Rules

The experiments presented here use a data set of 100 documents paired with filled templates. The average docuemnt length is over 200 words. We did a ten-fold cross-validation, dividing the data into 10 distinct testing sets and training on the remaining 90 documents. To evaluate the performance of the system with varying amounts of training data, we also ran tests with smaller subsets of the training examples for each test set and produced learning curves. Tests of machine learning systems usually measure simple accuracy: the number of examples that are correctly classified. In this type of task, however, since we don't have a set number of examples to be classified, simple accuracy has no clear meaning. There are really two measures which are important: precision, which is the percentage of the slot fillers which the system finds which are correct, and recall, which is the percentage of the slot fillers in the correct templates which are found by the system. If both precision and recall are 100%, then the results are completely correct. Lower precision indicates that the system is producing spurious fillers: that its rules are overly general. Lower recall indicates that the system is failing to find correct fillers: that its rules are too specific. Recent MUC conferences have introduced an F-measure [ARPA, 1992], combining precision and recall in order to provide a single number measurement for IE systems. We report the precision, recall, and F-measure with precision and recall weighted equally. For these experiments, we used the default values for all parameters

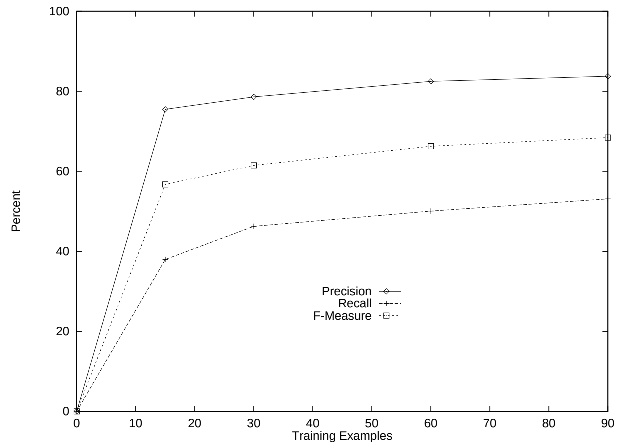

Figure 4: Performance on job postings

of the RAPIER algorithm: a beam-width of 10, stopping after 5 failures to compress, and abandoning specialization after 3 specialization iterations fail to produce a new best rule.

Figure 4 shows the learning curves generated. At 90 training examples, the average precision was 83.7% and the average recall was 53.1%. These numbers look quite promising when compared to the measured performance of other IE systems on various domains [Soderland *et al.*, 1995; Riloff, 1996; ARPA, 1992; 1993]. These comparisons are general, since the tasks are different, but they do indicate that RAPIER is doing relatively well.

It should be noted that the precision is close to 80% even with only 15 example documents. The "bottom-up" nature of the algorithm, coupled with the fact that the algorithm does not allow coverage of negatives, encourage it to create fairly specific rules, leading to this high precision. While the recall is less encouraging, it is likely that recall with continue to improve as the number of training examples increases.

5 Conclusion

Although ILP has advantages over propositional learning algorithms, its representation (first order logic) is not appropriate for some tasks which, because of their complexity, do require relational learning algorithm. We have developed a representation and a relational learning algorithm for one of these tasks, natural language information extraction. The success of this ILP-based algorithm on an alternate representation again demonstrates the utility of ILP research in areas far outside logic programming.

References

[ARPA, 1992] ARPA, editor. *Proceedings of the Fourth DARPA Message Understanding Evaluation and Conference*, San Mateo, CA, 1992. Morgan Kaufman.

[ARPA, 1993] ARPA, editor. *Proceedings of the Fifth DARPA Message Understanding Evaluation and Conference*, San Mateo, CA, 1993. Morgan Kaufman.

[Blockeel and deRaedt, 1996] Henrik Blockeel and Luc deRaedt. Relational knowledge discovery in databases. In *Proceedings of the Sixth International Workshop on Inductive Logic Programming"*, pages 1–13, 1996.

[Brill, 1994] Eric Brill. Some advances in rule-based part of speech tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 1994.

[Cohen, 1995] W. W. Cohen. Text categorization and relational learning. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 124–132, San Francisco, CA, 1995. Morgan Kaufman.

[Cohen, 1996] W. W. Cohen. Learning rules that classify e-mail. In *Papers from the AAAI Spring Symposium on Machine Learning in Information Access*, pages 18–25. AAAI Press, 1996.

[Huffmann, 1996] Scott B. Huffmann. Learning information extraction patterns from examples. In Stefan Wermter, Ellen Riloff, and Gabriele Scheller, editors, *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*, Lecture Notes in Artificial Intelligence, pages 246–260. Springer, 1996.

[Lehnert and Sundheim, 1991] Wendy Lehnert and Beth Sundheim. A performance evaluation of text-analysis technologies. *AI Magazine*, 12(3):81–94, 1991.

[Miller *et al.*, 1993] G. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. Introduction to WordNet: An on-line lexical database. *Available by ftp to clarity.princeton.edu*, 1993.

[Mooney and Califf, 1995] R. J. Mooney and M. E. Califf. Induction of first-order decision lists: Results on learning the past tense of English verbs. *Journal of Artificial Intelligence Research*, 3:1–24, 1995.

[Muggleton and Feng, 1992] S. Muggleton and C. Feng. Efficient induction of logic programs. In S. Muggleton, editor, *Inductive Logic Programming*, pages 281–297. Academic Press, New York, 1992.

[Muggleton, 1995] Steve Muggleton. Inverse entailment and Progol. *New Generation Computing Journal*, 13:245–286, 1995.

[Plotkin, 1970] G. D. Plotkin. A note on inductive generalization. In B. Meltzer and D. Michie, editors, *Machine Intelligence (Vol. 5)*. Elsevier North-Holland, New York, 1970.

[Quinlan, 1990] J.R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5(3):239–266, 1990.

[Riloff, 1996] Ellen Riloff. Automatically generating extraction patterns from untagged text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1044–1049, 1996.

[Soderland *et al.*, 1995] Stephen Soderland, D. Fisher, J. Aseltine, and W. Lehnert. Crystal: Inducing a conceptual dictionary. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1314–1319, 1995.

[Weizenbaum, 1966] J. Weizenbaum. ELIZA – A computer program for the study of natural language communications between men and machines. *Communications of the Association for Computing Machinery*, 9:36–45, 1966.

[Zelle and Mooney, 1994] J. M. Zelle and R. J. Mooney. Combining top-down and bottom-up methods in inductive logic programming. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 343–351, New Brunswick, NJ, July 1994.