Ph.D. Proposal, Department of Computer Sciences,
University of Texas at Austin, 1997

Relational Learning Techniques for Natural Language Information Extraction

Mary Elaine Califf
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712
mecaliff@cs.utexas.edu

Abstract

The recent growth of online information available in the form of natural language documents creates a greater need for computing systems with the ability to process those documents to simplify access to the information. One type of processing appropriate for many tasks is information extraction, a type of text skimming that retrieves specific types of information from text. Although information extraction systems have existed for two decades, these systems have generally been built by hand and contain domain specific information, making them difficult to port to other domains. A few researchers have begun to apply machine learning to information extraction tasks, but most of this work has involved applying learning to pieces of a much larger system. This paper presents a novel rule representation specific to natural language and a learning system, RAPIER, which learns information extraction rules. RAPIER takes pairs of documents and filled templates indicating the information to be extracted and learns patterns to extract fillers for the slots in the template. This proposal presents initial results on a small corpus of computer-related job postings with a preliminary version of RAPIER. Future research will involve several enhancements to RAPIER as well as more thorough testing on several domains and extension to additional natural language processing tasks. We intend to extend the rule representation and algorithm to allow for more types of constraints than are currently supported. We also plan to incorporate active learning, or sample selection, methods, specifically query by committee, into RAPIER. These methods have the potential to substantially reduce the amount of annotation required. We will explore the issue of distinguishing relevant and irrelevant messages, since currently RAPIER only extracts from the any messages given to it, assuming that all are relevant. We also intend to run much larger tests with RAPIER on multiple domains including the terrorism domain from the third and fourth Message Uncderstanding Conferences, which will allow comparison against other systems. Finally, we plan to demonstrate the generality of RAPIER's representation and algorithm by applying it to other natural language processing tasks such as word sense disambiguation.

1 Introduction

There has been an explosive growth in the amount of information available on networked computers around the world, much of it in the form of natural language documents. An increasing variety of search engines exist for retrieving such documents using keywords; however, answering many questions about available information requires a deeper "understanding" of natural language. One way of providing more "understanding" is with *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 MUC program (Lehnert & Sundheim, 1991). The extracted information can then be stored in a database which could then be queried using either standard database query languages or a natural language database interface. However, a difficulty with information extraction systems is that they are difficult and time-consuming to build, and they generally contain highly domain-specific components, making porting to new domains also time-consuming. Thus, more efficient means for developing information extraction systems are desirable.

Recent research in computational linguistics indicates that *empirical* or *corpus-based* methods are currently the most promising approach to developing robust, efficient natural language processing (NLP) systems (Church & Mercer, 1993; Charniak, 1993; Brill & Church, 1996). These methods automate the acquisition of much of the complex knowledge required for NLP by training on suitably annotated natural language corpora, e.g. *treebanks* of parsed sentences (Marcus, Santorini, & Marcinkiewicz, 1993).

Most of these empirical NLP methods employ statistical techniques such as *n-gram models*, *hidden Markov models* (HMMs), and *probabilistic context free grammars* (PCFGs). There has also been significant research applying neural-network methods to language processing (Reilly & Sharkey, 1992; Miikkulainen, 1993). However, there has been relatively little recent language research using symbolic learning, although some recent systems have successfully employed decision trees (Magerman, 1995; Anoe & Bennett, 1995), transformation rules (Brill, 1993, 1995), and other symbolic methods (Wermter, Riloff, & Scheler, 1996).

Given the successes of empirical NLP methods, researchers have recently begun to apply learning methods to the construction of information extraction systems (McCarthy & Lehnert, 1995; Soderland, Fisher, Aseltine, & Lehnert, 1995, 1996; Riloff, 1993, 1996; Kim & Moldovan, 1995; Huffmann, 1996). Several different symbolic and statistical methods have been employed, but most of them are used to generate one part of a larger information extraction system. Our system RAPIER (Robust Automated Production of Information Extraction Rules) learns rules for the complete information extraction task, rules producing the desired information pieces directly from the documents without prior parsing or any post-processing. We do this by using a structured (relational) symbolic representation, rather than learning classifications.

Using only a corpus of documents paired with filled template, RAPIER learns Eliza-like patterns (Weizenbaum, 1966) that make use of limited syntactic and semantic information, using freely available, robust knowledge sources such as a part-of-speech tagger or a lexicon. The rules built from these patterns can consider an unbounded context, giving them an advantage over more limited representations. This relatively rich representation requires a

learning algorithm capable of dealing with its complexities. Therefore, RAPIER employs a relational learning algorithm which uses techniques from several Inductive Logic Programming (ILP) systems. These techniques are appropriate because they were developed to work on a rich, relational representation (first-order logic clauses). Our algorithm incorporates ideas from several ILP systems, and consists primarily of a specific to general (bottom-up) search.

In this paper, we present preliminary experimental results with RAPIER on a small corpus of computer-related job postings. The precision and recall are comparable to those reported for information extraction systems operating in a variety of domains. We believe these results indicate that this approach shows considerable promise. We plan to test the algorithm more thoroughly on multiple information extraction tasks and to apply the system to other areas of natural language processing.

The remainder of this proposal is organized as follows. Section 2 presents background material on information extraction and learning. Section 3 describes the rule representation and RAPIER's learning algorithm. Section 4 presents and analyzes preliminary results obtained with RAPIER. In Section 5, we outline our future research plan, including extensions to RAPIER and plans for extensive testing in multiple domains. Section 6 discusses related work in applying learning to information extraction tasks. The final section discusses the goals and significance of this research.

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 & Sundheim, 1991; ARPA, 1992, 1993). Usually the data to be extracted is described by a template specifying a list of slots to be filled, though sometimes it is specified by annotations in the document. In either case, slot-fillers may be of two types: they may be one of a set of specified values or they may be strings taken directly from the document. Figures 1 and 2 show paired documents and templates from information extraction tasks in two very different domains. The job posting template includes only slots that are filled by strings taken directly from the document, while the Latin American terrorism template includes slots of both types.

Information extraction can be useful in a variety of domains. The various MUC's have focused on tasks such as Latin American terrorism, joint ventures, microelectronics, and company management changes. Others have used information extraction to track medical patient records (Soderland et al., 1995) and to track company mergers (Huffmann, 1996). Another domain which seems appropriate, particularly in the light of dealing with the wealth of online information, is to extract information from text documents in order to create easily searchable databases from the information, thus making the wealth of text online more easily accessible. For instance, information extracted from job postings in USENET newsgroups such as `misc.jobs.offered` can be used to create an easily searchable database of jobs. Such

Posting from Newsgroup

```
Subject: US-TN-SOFTWARE PROGRAMMER
Date: 17 Nov 1996 17:37:29 GMT
Organization: Reference.Com Posting Service
Message-ID: <56nigp$mrs@bilbo.reference.com>

SOFTWARE PROGRAMMER

Position available for Software Programmer experienced in generating
software for PC-Based Voice Mail systems.  Experienced in C Programming.
Must be familiar with communicating with and controlling voice cards;
preferable Dialogic, however, experience with others such as Rhetorix
and Natural Microsystems is okay. Prefer 5 years or more experience
with PC Based Voice Mail, but will consider as little as 2 years.
Need to find a Senior level person who can come on board and pick up
code with very little training. Present Operating System is DOS.
May go to OS-2 or UNIX in future.

Please reply to:
Kim Anderson
AdNET
(901) 458-2888 fax
kimander@memphisonline.com
```

Filled Template

```
computer_science_job
id: 56nigp$mrs@bilbo.reference.com
title: SOFTWARE PROGRAMMER
salary:
company:
recruiter:
state: TN
city:
country: US
language: C
platform: PC \ DOS \ OS-2 \ UNIX
application:
area: Voice Mail
req_years_experience: 2
desired_years_experience: 5
req_degree:
desired_degree:
post_date:  17 Nov 1996
```

Figure 1: Sample Message and Filled Template from the Job Posting Domain

Document

```
DEV-MUC3-0011 (NOSC)

   LIMA, 9 JAN 90 (EFE) -- [TEXT] AUTHORITIES HAVE REPORTED THAT FORMER
PERUVIAN DEFENSE MINISTER GENERAL ENRIQUE LOPEZ ALBUJAR DIED TODAY IN LIMA
AS A CONSEQUENCE OF A TERRORIST ATTACK.

   LOPEZ ALBUJAR, FORMER ARMY COMMANDER GENERAL AND DEFENSE MINISTER UNTIL
MAY 1989, WAS RIDDLED WITH BULLETS BY THREE YOUNG INDIVIDUALS AS HE WAS
GETTING OUT OF HIS CAR IN AN OPEN PARKING LOT IN A COMMERCIAL CENTER IN THE
RESIDENTIAL NEIGHBORHOOD OF SAN ISIDRO.

   LOPEZ ALBUJAR, 63, WAS DRIVING HIS OWN CAR WITHOUT AN ESCORT.  HE WAS SHOT
EIGHT TIMES IN THE CHEST.  THE FORMER MINISTER WAS RUSHED TO THE AIR FORCE
HOSPITAL WHERE HE DIED.
```

Filled Template

```
0.  MESSAGE: ID                     DEV-MUC3-0011 (NCCOSC)
1.  MESSAGE: TEMPLATE               1
2.  INCIDENT: DATE                  09 JAN 90
3.  INCIDENT: LOCATION              PERU: LIMA (CITY): SAN ISIDRO (NEIGHBORHOOD)
4.  INCIDENT: TYPE                  ATTACK
5.  INCIDENT: STAGE OF EXECUTION    ACCOMPLISHED
6.  INCIDENT: INSTRUMENT ID         -
7.  INCIDENT: INSTRUMENT TYPE       GUN: ‘‘-’’
8.  PERP: INCIDENT CATEGORY         -
9.  PERP: INDIVIDUAL ID             ‘‘THREE YOUNG INDIVIDUALS’’
10. PERP: ORGANIZATION ID           -
11. PERP: ORGANIZATION CONFIDENCE   -
12. PHYS TGT: ID                    -
13. PHYS TGT: TYPE                  -
14. PHYS TGT: NUMBER                -
15. PHYS TGT: FOREIGN NATION        -
16. PHYS TGT: EFFECT OF INCIDENT    -
17. PHYS TGT: TOTAL NUMBER          -
18. HUM TGT: NAME                   ‘‘ENRIQUE LOPEZ ALBUJAR’’
19. HUM TGT: DESCRIPTION            ‘‘FORMER ARMY COMMANDER GENERAL AND DEFENSE MI
NISTER’’: ‘‘ENRIQUE LOPEZ ALBUJAR’’
20. HUM TGT: TYPE                   FORMER GOVERNMENT OFFICIAL / FORMER ACTIVE MIL
ITARY: ‘‘ENRIQUE LOPEZ ALBUJAR’’
21. HUM TGT: NUMBER                 1: ‘‘ENRIQUE LOPEZ ALBUJAR’’
22. HUM TGT: FOREIGN NATION         -
23. HUM TGT: EFFECT OF INCIDENT     DEATH: ‘‘ENRIQUE LOPEZ ALBUJAR’’
24. HUM TGT: TOTAL NUMBER           -
```

Figure 2: Sample Message and Filled Template from the MUC Terrorism Domain

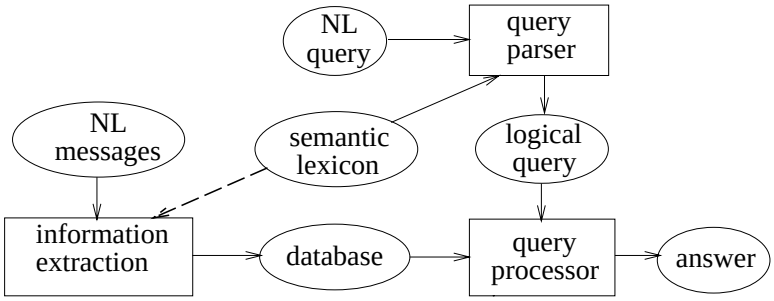

Figure 3: Complete System Architecture

databases would be particularly useful as part of a complete NLP system which supported natural language querying of the system. The architecture for such a complete system is shown in figure 3. A query parser can be learned using CHILL (Zelle & Mooney, 1996), a system which learns parsers from example sentences paired with their parses, and a semantic lexicon could be produced using WOLFIE (Thompson, 1995), a system which learns a lexicon from sentences paired with their semantic representations.

Information extraction systems are generally complex, with several modules, some of which are very domain specific. They usually incorporate parsers, specialized lexicons, and discourse processing modules to handle issues such as coreference. Most information extraction systems are built entirely by hand, though a few have incorporated learning in some modules.

2.2 Relational Learning

2.2.1 Why Symbolic Relational Learning?

Since most empirical work in natural language processing has employed statistical techniques, this section discusses the potential advantages of symbolic relational learning. In order to accurately estimate probabilities from limited data, most statistical techniques base their decisions on a very limited context, such as bigrams or trigrams (2 or 3 word contexts). However, NLP decisions must frequently be based on much larger contexts that include a variety of syntactic, semantic, and pragmatic cues. Consequently, researchers have begun to employ learning techniques that can handle larger contexts, such as *decision trees* (Magerman, 1995; Miller, Stallard, Bobrow, & Schwartz, 1996; Anoe & Bennett, 1995) or *exemplar* (*case-based*) methods (Cardie, 1993; Ng & Lee, 1996). However, these techniques still require the system developer to specify a manageable, finite set of features for use in making decisions. Developing this set of features can require significant representation engineering and may still exclude important contextual information.

In contrast, *relational learning* methods (Birnbaum & Collins, 1991) allow induction over *structured* examples that can include first-order logical predicates and functions and unbounded data structures such as lists and trees. In particular, *inductive logic programming* (ILP) (Lavrač & Džeroski, 1994; Muggleton, 1992) studies the induction of rules in first-order logic (Prolog programs). ILP systems have induced a variety of basic Prolog programs

(e.g. `append, reverse, sort`) as well as potentially useful rule bases for important biological problems (Muggleton, King, & Sternberg, 1992; Srinivasan, Muggleton, Sternberg, & King, 1996). 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 & Califf, 1995).

Two other advantages of ILP-based techniques are comprehensibility and the ability to use background knowledge. 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). With respect to background knowledge, ILP systems are given Prolog definitions for a set of predicates that can be used in the body of learned rules.

While RAPIER is not an ILP system, it is a relational learning algorithm learning a structured rule representation, and its algorithm was inspired by ideas from ILP systems. The ILP-based ideas are appropriate because they were designed to with rich, unbounded representations. The following sections briefly describe the three systems which most directly influenced RAPIER's algorithm.

2.2.2 GOLEM

GOLEM (Muggleton & Feng, 1992) is a bottom-up (specific to general) ILP algorithm based on the construction of relative least-general generalizations, *rlggs* (Plotkin, 1970). The idea of least-general generalizations (LGGs) is, given two items (in ILP, two clauses), finding the least general item that covers the original pair. This is usually a fairly simple computation. *Rlggs* are the LGGs *relative* to a set of background relations. Because of the difficulties introduced by non-finite *rlggs*, background predicates must be defined extensionally. The algorithm operates by randomly selecting several pairs of positive examples and computing the determinate *rlggs* of each pair. Determinacy constrains the clause to have for each example no more than one possible valid substitution for each variable in the body of the clause. The resulting clause with the greatest coverage of positive examples is selected, and that clause is further generalized by computing the *rlggs* of the selected clause with new randomly chosen positive examples. The generalization process stops when the coverage of the best clause no longer increases.

2.2.3 CHILLIN

The CHILLIN (Zelle & 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.

2.2.4 PROGOL

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 a 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.

2.3 NLP Resources

Part of the purpose of this research is to make use of the available resources for natural language processing to provide syntactic and semantic information rather than spending human time developing specialized parsers or lexicons. We plan to use two primary types of existing systems. For basic syntactic information we are employing a part-of-speech tagger, a program which takes sentences as input and labels each word or symbol in the sentence with a part-of-speech tag (eg. noun, verb, adjective, preposition). This doesn't provide as much information as a parser, but a tagger is faster and more robust than a full parser. At present we are using Eric Brill's tagger as trained on a Wall Street Journal corpus (Brill, 1994). One advantage to this particular tagger is that it can be trained on a new domain to improve performance. For semantic information, we plan to employ a domain-independent lexicon which includes a semantic hierarchy, possibly supplemented by domain specific lexicons. Initially, we plan to use WordNet (Miller, Beckwith, Fellbaum, Gross, & Miller, 1993), a lexical database of over 50,000 words which contains a semantic hierarchy in the form of *hypernym* links.

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

Figure 4: A Rule Extracting an Area Filler

3 RAPIER Algorithm

3.1 Rule Representation

RAPIER's rule representation uses Eliza-like patterns (Weizenbaum, 1966) that make use of limited syntactic and semantic information. For the initial implementation, we did 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 template name and slot name and consist of three parts: 1) a pre-filler pattern that matches 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; or the negation of each of the above: lists of words, tags, or semantic classes, none of may match the document item. Figure 4 shows a rule created by hand that extracts the `area` filler (telecommunications) from an example document containing a phrase such as "Leading telecommunications firm in need of . . .". The values `nn` and `nns` for the syntactic constraint indicate that the matching words must be tagged as a common noun or a plural common noun.

3.2 Learning Algorithm

RAPIER, as noted above, is inspired by ILP methods, particularly by GOLEM, CHILLIN, and PROGOL, and primarily consists of a specific to general (bottom-up) search. Like CHILLIN, RAPIER begins with a most specific definition and then attempts to compact that definition by replacing rules by more general rules. Therefore, for each slot, most-specific patterns are created from each filler for that slot in each example, specifying word and tag for the filler and its context, the context being the entire document. 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 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 covering negatives, so a standard approach would be to generate the 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):

$$ruleVal = -log_2(p/(p+n)) + ruleSize/10$$

where p is the number of correct fillers extracted by the rule. Because in this type of task there are no provided negative examples, we use a notion of implicit negatives and count the number of spurious fillers a rule extracts from the training examples as n. The size of the rule is computed using a simple heuristic, counting 1 for each constraint disjunct, 2 for each pattern item, and 3 for each pattern list. The rule size is divided by 10 simply to keep the values commensurate with the information value since the purpose of the size component is simply to encourage the system to prefer more general rules, especially to prefer the smaller rule in the case where two rules have the same information value. 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 5 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.

As an example of the creation of a new rule, consider generalizing the rules based on the phrases "located in Atlanta, Georgia." and "offices in Kansas City, Missouri." The rules created from these phrases for the city slot would be

```
Initialization
    AllRules = most specific rules from example documents
For each slot, S in the template being learned
    SlotRules = rules in AllRules 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 them, and initialize RuleList
      let n = 0
      while best rule in RuleList produces spurious fillers and the weighted
            information value of the best rule is improving
        increment n
        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 of RulesList
      if best rule in RuleList produces only valid fillers
        Add it to SlotRules and AllRules and remove empirically subsumed rules
```

Figure 5: RAPIER Algorithm for Inducing Information Extraction Rules

Pre-filler Pattern:	Filler Pattern:	Post-filler Pattern:
1) word: located tag: vbn 2) word: in tag: in	1) word: atlanta tag: nnp	1) word: , tag: , 2) word: georgia tag: nnp 3) word: . tag: .

and

Pre-filler Pattern:	Filler Pattern:	Post-filler Pattern:
1) word: offices tag: nns 2) word: in tag: in	1) word: kansas tag: nnp 2) word: city tag: nnp	1) word: , tag: , 2) word: missouri tag: nnp 3) word: . tag: .

The fillers are generalized to produce two possible rules with empty pre-filler and post-filler patterns. Because one filler has two items and the other only one, they generalize to a list of no more than two words. The word constraints generalize to either a disjunction of all the words or no constraint. The tag constraints on all of the items are the same, so the generalized rule's tag constraints are also the same. Since the three words do not belong to a single semantic class in the lexicon, the semantics remain unconstrained. The fillers produced are:

Pre-filler Pattern:	Filler Pattern:	Post-filler Pattern:
	1) list: max length: 2 word: [atlanta, kansas, city] tag: nnp	

and

Pre-filler Pattern:	Filler Pattern:	Post-filler Pattern:
	1) list: max length: 2 tag: nnp	

Either of these rules is likely to cover spurious examples, so we add pre-filler and post-filler generalizations. The items produced from the "in"'s and the commas are identical and, therefore, unchanged. Assuming that our lexicon contains a semantic class for states, generalizing the state names produces a semantic constraint of that class along with a tag constraint nnp and either no word constraint or the disjunction of the two states. Thus, a final best rule would be:

Pre-filler Pattern:	Filler Pattern:	Post-filler Pattern:
1) word: in tag: in	1) list: max length: 2 tag: nnp	1) word: , tag: , 2) tag: nnp semantic: state

4 Experimental Evaluation

This section presents preliminary results obtained with the current version of RAPIER on computer-related job posting domain.

The task we have chosen for initial tests of RAPIER is to extract information from computer-related job postings that could be used to create a database of available jobs. The computer-related job posting 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. All of the slots take values that are strings taken directly the text, since RAPIER does not yet handle the other type of slots. The current results do not employ semantic categories, only words and the results of Brill's POS tagger. It also does not use negative constraints.

The experiments presented here use a data set of 100 documents paired with filled templates. The average document 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 information extraction 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 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 6 shows the performance of RAPIER on the test data. 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 information extraction systems on various domains. This performance is comparable to that of CRYSTAL on a medical domain task (Soderland et al., 1996), and better than that of AUTOSLOG and AUTOSLOG-TS on part of the MUC4 terrorism task (Riloff, 1996). It also compares favorably with the typical system performance on the MUC tasks (ARPA, 1992, 1993). All of these comparisons are only 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

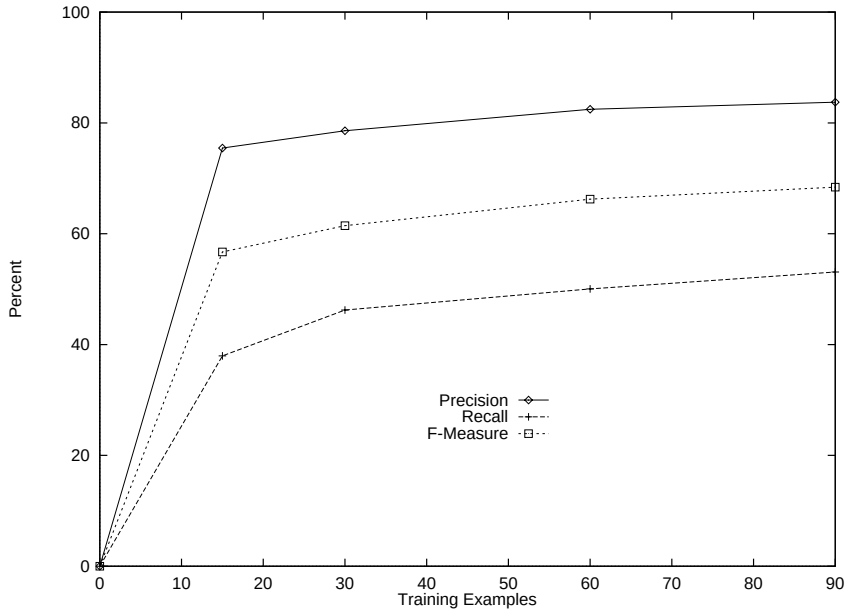

Figure 6: Performance on job postings

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.

The rules RAPIER learns are of several different types. Some are fairly simple memorizations of words or phrases that consistently appear in particular slots: these include things like programming languages and operating systems. For example, `AIX` and `MVS` always fill the platform slot when they appear, so one rule for that slot is:

Pre-filler Pattern:	Filler Pattern:	Post-filler Pattern:
	1) word: [aix, mvs] tag: nnp	

This constrains the filler to be either aix or mvs, tagged as a proper noun. Some rules memorize particular words or phrases but also require them to be in specific contexts, because the phrase may appear in circumstances where it is not a valid filler.

Clearly, mere memorization of words and phrases is seldom interesting, and other rules learn the context of the filler, usually also constraining the parts of speech of the filler. One example of this is the following rule for extracting slot fillers for area:

Pre-filler Pattern:	Filler Pattern:	Post-filler Pattern:
1) word: [knowledge, areas]	1) list: max length: 3	1) word: [technology,',’]
2) tag: in	tag: [nnp, nn]	2) tag: [endsent, nnp]

`endsent` is a token added by the system to denote the ends of sentences (or other distinct fragments of text) since these may not be denoted by punctuation in netnews postings.

5 Future Research

We plan to address a number of issues in future research. These fall into three primary areas. First there are several enhancements which we plan to make to RAPIER's algorithm. We also plan to incorporate active learning into the RAPIER system in order to reduce both the work involved in tagging corpora and the amount of training data needed to achieve acceptable levels of performance. Next, we wish to do far more extensive evaluations of RAPIER on both the job postings domain and additional domains. Finally, we intend to explore the applicability of RAPIER's representation and basic algorithm to other natural language processing tasks. Each of these areas of future research is discussed in some detail below.

5.1 Enhancing RAPIER

The version of RAPIER described thus far is a preliminary implementation, and we plan several enhancements. First, an examination of the rules produced by RAPIER indicates that the decision to select a single random pair at a time from which to produce generalizations was probably unwise. This was done because each pair of rules can generate a large number of potential generalizations. However, while some rules produced by the system are clearly good generalizations covering a number of examples, others are simply disjunctions of completely unrelated items. As an example, one rule produced the following rule for the `area` slot:

Pre-filler Pattern:	Filler Pattern:	Post-filler Pattern:
1) word: [experience, +]	1) list: max length: 3	1) word: [and, oop]
2) tag: [endsent, sym]	: [:, nnp, njj, n]	2) word: [data, rt]

Clearly, this rule is the result of generalizing two fairly unrelated examples, and it is not an intuitively good generalization. Studies of traces of the algorithm show that a lot of time and effort is wasted exploring pairs of rules that have no hope of producing a useful new rule. Therefore, we intend to modify the algorithm to be more similar to GOLEM and CHILLIN, starting with several randomly selected pairs of rules instead of a single pair. This should help the algorithm to find more useful pairs of rules to generalize.

Second, although mechanisms for dealing with semantic constraints are present in the system, we have not yet implemented an interface to a lexicon to provide the semantic information needed to make use of semantic constraints. We plan to connect the system to WordNet initially, since this is a freely available lexicon which can provide the kind of semantic information we're looking for. We may also use domain-specific sources of semantic information such as dictionaries of places, companies, programming languages, etc. Besides using a lexicon to introduce semantic constraints, we plan to allow the algorithm to create semantic classes. If a single pattern item had a word constraint consisting of the disjunction of a long list of words, a new semantic class would be created whose member words would consist of the words in the word constraint, and the constraint would be replaced by a semantic constraint. This semantic class would then be available to the system as it generalized other rules. The development of this "new" background knowledge is analogous to predicate

invention in ILP (Kijsirikul, Numao, & Shimura, 1992; Zelle & Mooney, 1994). It would be interesting also to attempt to incorporate new words into an existing semantic class. For example, we might have a semantic class consisting of programming languages which we would like to expand automatically to incorporate new languages.

Another addition we plan for the algorithm is to add negative constraints: that is words, tags, or semantic classes that must not appear in the text matching the item or list with the constraint. These constraints would be found by comparing correct fillers found by a rule with the spurious fillers it produces and producing constraints from elements of the spurious fillers. Constraints the match a large percentage of the spurious fillers, but none, or few, of the correct fillers would be added to the rule, and the resulting new rule(s) would be added to the list of considered rules.

We may also extend RAPIER's rule representation in other ways. One possibility that we've considered is incorporating morphological information. To do this, we would use the morphological functions in WordNet and then allow for constraints on the roots and inflections of words as well as on the surface form of the word. We may also consider using some form of syntactic parsing eventually. In order to use this information, we can simply use pattern elements that are syntactic phrases such as "noun phrase," "subject" or "prepositional phrase," along with the single word and arbitrary lists that the representation currently employs.

We also plan to explore alternate rule evaluation metrics. The current metric for evaluating rules is based on the information value of the rule penalized by the size of the rule. We would like to explore variations on this metric. We would also like to explore metrics based on the minimum description length (MDL) principle(Quinlan & Rivest, 1989). The idea behind MDL is that the "best" generalization is one that minimizes the size of the description of the data. Thus, in developing a theory that describes data, an MDL metric seeks to minimize the size of the theory plus the corrections to the theory to handle examples not correctly handled by the theory (usually simply the examples not yet covered). The difficulty of using MDL is that we will have to determine how to calculate the size of corrections for spurious fillers produced. We wish to determine empirically whether any of these metrics seems to have an advantage over the others in the domains we're examining. We may also explore the possibility of allowing for noise and accepting rules which produce a few spurious fillers if they also cover a large number of postives. One advantage of the MDL metric is that it does allow for noise handling, but we may also want to look at other metrics which are better for handling noise (Lavrač & Džeroski, 1994).

There are also several parameters to the system which should be empirically tuned. These parameters include the width of the beam in the beam search, the number of failures to improve the best rule in the beam allowed before the search is terminated, and the number of failures to find a new rule to compress the rule base before compression of rules for a slot is abandoned. In all these cases, increasing the size of the parameter might lead to better rule bases and will lead to longer training times. The values used in the trials presented here have been simply choices that seemed reasonable default values, but these parameters have not been tested to see where the ideal tradeoff seems to fall.

As discussed above, the current RAPIER system only extracts values that are strings taken directly from the original document text. However, for some tasks, it may be more appropriate to select one of a pre-specified set of values. Several slots in the terrorism task template are of this type: the type of incident, the stage of execution of the incident (accomplished, attempted, or threatened), the category of instrument used (gun, bomb, grenade, etc.), and the category of the physical target, among others. It should be possible to extend the algorithm to handle extracting values of this type. Changes to the system itself should be relatively straightforward. First, the rule representation would need to be modified to include the value to be extracted, which would be either the value from the set in the case of the pre-defined set of values or an indication that the string matching the filler pattern is the value to be extracted (the case handled currently). Second, the generalization process would need to be modified slightly to take into account the value to extracted. Clearly the only interesting pairs of rules to generalize are those which extract the same value or which both extract a string from the document. Finally, the training data provided to the system would need some extension. Besides the actual value to be extracted, the system needs to know what portion of the document it should create patterns from to extract that value. In the case of strings taken from the document, this is very straightforward: we simply use all occurrences of the string in the document to anchor rules. In other cases, however, the value and the portion of the document which should anchor the rule must both be specified. For example, a rule that identifies the stage of execution of a terrorist incident as "accomplished" might be anchored by the phrase "HAVE BEEN KIDNAPPED" or "THE MASS KIDNAPPING TOOK PLACE" since these are points in the document which indicate that the incident was, in fact, accomplished rather than merely attempted or threatened. The anchor is necessary to provide the RAPIER heuristics a starting point to work from. We may, however, explore the possibility of finding words or phrases common to documents sharing a particular slot value but rare in other documents as AUTOSLOG-TS does(Riloff, 1996), and then using those words or phrases as anchors for rules.

A final issue which we need to address in the RAPIER system is that of identifying relevant documents. In the experiments presented here, all of the documents for training and testing are relevant for the task, i.e. all are job postings about a computer-related job. However, a fully automated system will have to deal with both relevant and irrelevant documents, or with choosing the relevant template to use: eg. given a job posting, the system might need to categorize it as a computer-related job, an engineering job, some other kind of job posting, or an irrelevant post.

The RAPIER system as it now stands does not address this issue, but we have considered a few different alternatives as to how to deal with it. One option is to run each document though RAPIER's extraction rules for all possible templates and then use the results of the extraction to determine which template, if any, is appropriate for the document. Presumably the most relevant template would have the highest percentage of slots filled, and documents which are completely irrelevant should have very few slots filled. However, the system would probably need to learn which slots were actually useful in making the relevance decisions. Some slots might be highly specific to a particular template, while others might be easily filled even for irrelevant documents. For example, the date a job offering was posted is useful information to extract, but it is information that will be extracted for any netnews post.

Being able to extract a filler for the number of years of experience required is strong evidence that the message is a job posting, but does nothing to distinguish a computer job from any other kind of job.

A second option for dealing with the issue of distinguishing between relevant and irrelevant documents and for determining which of a set of templates is most appropriate would be to use standard information retrieval techniques (Salton, 1989; Frakes & Baeza-Yates, 1992) to classify the documents initially and then to pass the document on to the information extraction system, the decision as to which template to use having already been made.

Finally, we might try to extend RAPIER to handle the document classification task as a separate slot with fixed values. The primary difficulty here may be the issue of anchoring the rules, since it does not seem feasible to have a person identify a particular portion of each document that is the right place to anchor a rule determining the classification of the document. However, the idea of finding phrases that distinguish particular filler values and using that phrases to anchor rules, as described above, would alleviate this problem.

All of these options seem potentially viable, though all have weaknesses as well as strengths. Empirical tests will be required to determine what the best method of handling the classification issue is.

5.2 Active Learning

In addition to the above, we intend to use active learning techniques, specifically *selective sampling* (Cohn, Atlas, & Ladner, 1994; Lewis & Catlett, 1994; Dagan & Engelson, 1995), to reduce annotation effort. Not all training examples are equally useful to a learning system, and by carefully selecting useful examples, annotation effort can be dramatically reduced. This approach has been successfully used in training part-of-speech taggers (Engelson & Dagan, 1996), and we believe that it will be helpful in the information extraction domain, as well.

Active learning involves constructing or selecting informative training examples rather than passively accepting them. Since raw text is widely available, *sample selection*, choosing a subset of examples to annotate from an unanalyzed corpus, is the style of active learning most useful for natural language applications (Engelson & Dagan, 1996). In *committee-based sampling*, the learner maintains multiple definitions consistent with the current training data (i.e. multiple elements of the *version space* (Mitchell, 1982)), categorizes unlabelled examples with each definition, and selects for labelling those examples with the most disagreement amongst the "committee members" (Seung, Opper, & Sompolinsky, 1992; Cohn et al., 1994). By singling out only such potentially informative examples for annotation, equivalent accuracy can be gained from labelling significantly fewer examples compared to random selection.

We intend to apply selective sampling by using a version of Query by Committee (Seung et al., 1992):

> Initialize the committee to two hypotheses formed from some initial labelled data
> For each unlabelled example do
> If the two committee members disagree on its label

Then Request the label of the example, add it to the training set, a nd
Update the two hypotheses to be consistent with all current training data

Alternatively, a larger committee can be used and each example selected for annotation with a probability proportional to the amount of disagreement over its label (Engelson & Dagan, 1996). Since the learning algorithm is not incremental, retraining after each selected example may be too time consuming, so samples will probably need to be selected in batches (Lewis & Catlett, 1994).

Since RAPIER employs a random sampling of pairs as seeds to the bottom-up generalization algorithm, committee member can be generated by using different random samples. In order to more fairly sample the version space and avoid over-agreement, it may be preferable to encourage one hypothesis to be quite specific and the other to be quite general, as in (Cohn et al., 1994). Since the induction algorithm performs a combination of top-down and bottom-up search, it should be possible to alter these in various ways to control the generality of the learned rules. One option might be to require a certain level of compression of the rules for each slot and thus vary the level of compression in the two rule sets.

In information extraction, each slot can be treated separately since the user can be asked to provide only the values for some slots for a message. However, if a value for a slot is not found by any of the committee members, it may be because it appears in a novel context which is not recognized by any of the learned rules, as opposed to being truly absent. This is particularly likely given the tendency of the RAPIER algorithm to produce rules with high precision but relatively low recall. Consequently, even for examples where committee members agree there is no filler for a particular slot, we propose to probabilistically request the value of the slot according to the *a priori* probability that messages provide a value for this slot (as determined from the existing training data).

Besides using active learning to reduce annotation efforts, we propose to test the usefulness of the active learning techniques by comparing the performance of the system as trained on sets of randomly selected examples to the performance when selective sampling is used to build a training set of the same size. We believe that the use of active learning may be valuable for helping to raise recall more quickly. The learning curve shows that in the job postings domain, RAPIER clearly learns some very useful rules from only a few examples, but then other more sparsely represented cases require much more data. Active learning would eliminate the new examples that fit those common, easily learned cases and focus the learning on the less common ones.

5.3 Extension to Additional Domains

One of the important reasons for building an information extraction system which learns rules from data is that it should be easy to apply the system to multiple domains. Therefore, an important part of our research will be demonstrating that the system does work within multiple domains.

One way in which we will expand the testing of the system will be to incorporate additional types of jobs into the job postings database. Our intention is to create information

extraction templates for several additional types of jobs that are commonly posted, such as engineering jobs and also to create a general job template which would provide information for job postings that don't fall into any of the more specific categories.

Another domain we wish to examine is that of Latin American terrorism as used in the Third and Fourth Message Understanding Conferences. The data for this domain consist of 1700 documents plus the filled templates for those documents. About half of the documents are considered irrelevant for the task. There are 25 slots in the templates; seven of these are filled by taking strings directly from the original text. We plan to do experiments involving these seven slots initially, and then to experiment with the full templates after RAPIER is modified to handle other types of slot fillers. We feel that these experiments are important as an opportunity to compare the performance of an information extraction system which learns its rules to systems which were hand-built.

We also plan to look for additional domains to apply RAPIER to. We are considering additional types of newsgroups. We also believe that there may be appropriate applications involving web pages. For example, intelligent agents for searching the web might be looking for specific types of information (eg. products and their prices in the case of a shopping agent). Rules of the type learned by RAPIER may be appropriate for extracting this needed information from pages, in which case the system could be used to learn the rules.

5.4 Extension to Other Natural Language Processing Tasks

We believe that the representation used by RAPIER as well as the basic algorithm will also prove useful for natural language processing tasks other than information extraction. One such task is learning rules for word sense disambiguation. This task seems particularly appropriate because it can be easily mapped into the current system. A template would be created for the word to be disambiguated, with the template slots being the various senses of the word. The filler in all cases would be the word. The system would then learn pre-filler and post-filler patterns that would select the word only if it had the desired sense. Of course, the content of the filler would be the same in all rules for each word, and the interesting result of running the rules would be determining which sense of the word had a filler selected for it. An alternative way to map the word sense disambiguation task to the information extraction task would be to have the template consist of a single slot with a fixed set of values (the possible word senses). We intend to run some experiments with word disambiguation corpora that have been used with other learning systems (Mooney, 1996) to see whether RAPIER's representation and algorithm are successful at this task.

We also hope to find additional natural language processing tasks for which our representation and algorithm seem appropriate. One possibility would be text classification using patterns automatically anchored as described above for fixed-value slots. Good text classification may, however, require that multiple patterns anchored in different parts of the text be learned for a single rule.

6 Related Work

Previous researchers have generally applied machine learning only to pieces of the information extraction task and their systems have typically required more human interaction than just providing texts with filled templates. One of the earliest attempts to use learning in an information extraction system was AUTOSLOG (Riloff, 1993). AUTOSLOG creates a dictionary of extraction patterns by specializing a set of general syntactic patterns. These patterns are used by a larger information extraction system including a parser and a discourse analysis module. AUTOSLOG has two major drawbacks. First, it requires a human expert to examine the patterns that it produces to determine which should be kept in the extraction pattern dictionary. Thus, it is speeding up the construction of the dictionary, but not fully automating it. Second, the specialization of the set of general patterns is done by looking at one example at a time. It doesn't take into account the number of other correct examples the specialization might also cover, or the number of times in the sample data that the pattern could trigger incorrectly. This is one reason why a human expert is necessary to examine the validity of the patterns generated. A newer version–AUTOSLOG-TS (Riloff, 1996)–generates potentially useful patterns by using statistics about those patterns matching relevant and irrelevant documents. This system has better precision tha AUTOSLOG because it does rely on scoring patterns based on the number of times they appear in the documents, but it does not use templates or annotation at all. A human must determine which slot a given extraction pattern is for, and as with the earlier system a human must go through the generated patterns and select those that will actually be used by the system.

CRYSTAL (Soderland et al., 1995, 1996) is a more recent attempt to apply machine learning to the creation of information extraction patterns. Its training instances are created by a sentence analyzer which identifies syntactic constituents such as subject, verb, object and tags each word with a semantic class. CRYSTAL also requires a semantic hierarchy for the domain and a lists of concepts for the domain. Like AUTOSLOG, CRYSTAL's extraction patterns are syntactically based; the "concept definitions" consist of constraints on the syntactic constituents of an instance. The constraints can be on such things as words appearing in the constituent, semantic class, or the root of a verb. The definition also indicates what constituent(s) of the instance are to be extracted for what slots. CRYSTAL generalizes its initial concept definitions by taking a seed instance and relaxing the constraints on its constituents to cover additional instances that extract the same slots. Generalization ends when too many negative examples will be covered by further relaxation of the constraints.

Another system that learns information extraction patterns is PALKA (Kim & Moldovan, 1995). PALKA represents it rules as *FP-structures*, which constrain the root of the verb and have semantic constraints on the phrases to be extracted. It generalizes and specializes the rules by moving up and down in a semantic hierarchy or adding disjunctions of semantic classes. PALKA's representation is limited by its inability to place word constraints on noun or prepositional phrases, and by its failure to place semantic constraints on any noun or prepositional phrases that are not to be extracted. Like the previous systems, PALKA relies on prior sentence analysis to identify syntactic elements and their relationships.

LIEP (Huffmann, 1996) also learns information extraction patterns. In many ways, LIEP functions like AUTOSLOG except that it learns only patterns that extract multiple slots,

rather than a single slot per pattern. LIEP's extraction rules have syntactic constraints on pair-wise syntactic relationship between sentence elements. It finds the relationships that link elements to be extracted, adding non-extracted elements as need to form a path between the extracted elements. Extraction patterns also contain semantic constraints, but these are not generalized. The learning algorithm uses seed examples, proposing up to three rules from each example. LIEP tests each rule on the training set and keeps the one with the best F-measure. LIEP's primary limitations are that it also requires a sentence analyzer to identify noun groups, verbs, subjects, etc.; it makes little use of semantic information; and it assumes that all information it needs is between two entities to be extracted.

Two primary things distinguish RAPIER from other systems that learn extraction patterns. First, RAPIER is intended to handle the information extraction task on its own, rather than being a part of a larger information extraction system. All of these systems require a sentence analyzer, and most require later parts of the full information extraction system to clean up their output. CRYSTAL and PALKA require domain-specific semantic hierarchies. On the other hand, we use only freely available taggers and lexicons and do not do any post-processing of RAPIER's output.

The second distinguishing characteristic is RAPIER's learning algorithm. Of the systems described, only CRYSTAL uses generally applicable machine learning techniques to create generalizations of the training examples. AUTOSLOG uses a set of heuristics to create generalizations from single examples, but it does not evaluate those generalizations on the training set and thus makes no guarantees about the performance of its patterns on the training data. LIEP does evaluate the generalizations it proposes based on their coverage of the training examples, but it simply proposes three generalizations based on one example and picks the best of those; thus, it doesn't really make use of the other training examples in its generalization process. PALKA does use training examples to guide its generalization and specialization, but its algorithm is highly domain specific since generalization and specialization consist entirely in moving up and down within a domain-specific semantic hierarchy or adding disjunctions of semantic classes. RAPIER is based on general relational learning techniques, and while its representation is specific to domains that can be represented as strings, we believe that it will prove to be applicable to NLP tasks other than information extraction.

7 Conclusion

The ability to extract desired pieces of information from natural language texts is an important task with a growing number of potential applications. Tasks requiring locating specific data in newsgroup messages or web pages are particularly promising applications. Manually constructing such information extraction systems is a laborious task; however, learning methods have the potential to help automate the development process. The RAPIER system described in this proposal uses relational learning to construct unbounded pattern-match rules for information extraction given only a database of texts and filled templates. The learned patterns employ limited syntactic and semantic information to identify potential slot fillers and their surrounding context. Results on extracting information from newsgroup jobs postings have shown that for one realistic application, fairly accurate rules can

be learned from relatively small sets of examples. Future enhancements to RAPIER include improvements to the learning algorithm, the incorporation of more semantic information, and the incorporation of active learning techniques. RAPIER with be tested on additional information extraction tasks, including some in which its performance can be more directly compared to other information extraction systems. Finally, RAPIER will be applied to other areas of natural language processing.

References

Anoe, C., & Bennett, S. W. (1995). Evaluating automated and manual acquisition of anaphora resolution strategies. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Lenguistics*, pp. 122–129 Cambridge, MA.

ARPA (Ed.). (1992). *Proceedings of the Fourth DARPA Message Understanding Evaluation and Conference*, San Mateo, CA. Morgan Kaufman.

ARPA (Ed.). (1993). *Proceedings of the Fifth DARPA Message Understanding Evaluation and Conference*, San Mateo, CA. Morgan Kaufman.

Birnbaum, L. A., & Collins, G. C. (Eds.). (1991). *Proceedings of the Eighth International Workshop on Machine Learning: Part VI Learning Relations*, Evanston, IL.

Brill, E. (1993). Automatic grammar induction and parsing free text: A transformation-based approach. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pp. 259–265 Columbus, Ohio.

Brill, E. (1995). Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, *21*(4), 543–565.

Brill, E., & Church, K. (Eds.). (1996). *Proceedings of the Conference on Empirical Methods in Natural Language Processing.* University of Pennsylvania, Philadelphia, PA.

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

Cardie, C. (1993). A case-based apprach to knowledge acquisition for domain-specific sentence analysis. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pp. 798–803.

Charniak, E. (1993). *Statistical Language Learning.* MIT Press.

Church, K., & Mercer, R. L. (1993). Introduction to the special issue on computational linguistics using large corpora. *Computational Linguistics*, *19*(1), 1–24.

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

Cohen, W. W. (1996). Learning rules that classify e-mail. In *Papers from the AAAI Fall Symposium on AI Applications in Knowledge Navigation & Retrieval*, pp. 18–25. AAAI Press.

Cohn, D., Atlas, L., & Ladner, R. (1994). Improving generalization with active learning. *Machine Learning*, *15*(2), 201–221.

Dagan, I., & Engelson, S. P. (1995). Committee-based sampling for training probabilistic classifiers. In *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 150–157 San Francisco, CA. Morgan Kaufman.

Engelson, S., & Dagan, I. (1996). Minimizing manual annotation cost in supervised training from corpora. In *Proceedings of the 34th Annual Meeting of the Association for Computational Lenguistics* Santa Cruz, CA.

Frakes, W., & Baeza-Yates, R. (Eds.). (1992). *Information Retrieval: Data Structures and Algorithms*. Prentice Hall.

Huffmann, S. B. (1996). Learning information extraction patterns from examples. In Wermter, S., Riloff, E., & Scheller, G. (Eds.), *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*, Lecture Notes in Artificial Intelligence, pp. 246–260. Springer.

Kijsirikul, B., Numao, M., & Shimura, M. (1992). Discrimination-based constructive induction of logic programs. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pp. 44–49 San Jose, CA.

Kim, J.-T., & Moldovan, D. I. (1995). Acquisition of linguistic patterns for knowledge-based information extraction. *IEEE Transactions on Knowledge and DataEngineering*, *7*(5), 713–724.

Lavrač, N., & Džeroski, S. (1994). *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood.

Lehnert, W., & Sundheim, B. (1991). A performance evaluation of text-analysis technologies. *AI Magazine*, *12*(3), 81–94.

Lewis, D. D., & Catlett, J. (1994). Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 148–156 San Francisco, CA. Morgan Kaufman.

Magerman, D. M. (1995). Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Lenguistics*, pp. 276–283 Cambridge, MA.

Marcus, M., Santorini, B., & Marcinkiewicz, M. (1993). Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, *19*(2), 313–330.

McCarthy, J., & Lehnert, W. (1995). Using decision trees for coreference resolution. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp. 1050–1055.

Miikkulainen, R. (1993). *Subsymbolic Natural Language Processing: An Integrated Model of Scripts, Lexicon, and Memory*. MIT Press, Cambridge, MA.

Miller, G., Beckwith, R., Fellbaum, C., Gross, D., & Miller, K. (1993). Introduction to WordNet: An on-line lexical database. *Available by ftp to clarity.princeton.edu*.

Miller, S., Stallard, D., Bobrow, R., & Schwartz, R. (1996). A fully statistical approach to natural language interfaces. In *Proceedings of the 34th Annual Meeting of the Association for Computational Lenguistics*, pp. 55–61 Santa Cruz, CA.

Mitchell, T. M. (1982). Generalization as search. *Artificial Intelligence*, *18*(2), 203–226.

Mooney, R. J. (1996). Comparative experiments on disambiguating word senses: An illustration of the role of bias in machine learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 82–91 Philadelphia, PA.

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

Muggleton, S., & Feng, C. (1992). Efficient induction of logic programs. In Muggleton, S. (Ed.), *Inductive Logic Programming*, pp. 281–297. Academic Press, New York.

Muggleton, S., King, R., & Sternberg, M. (1992). Protein secondary structure prediction using logic-based machine learning. *Protein Engineering*, *5*(7), 647–657.

Muggleton, S. H. (Ed.). (1992). *Inductive Logic Programming*. Academic Press, New York, NY.

Muggleton, S. (1995). Inverse entailment and Progol. *New Generation Computing Journal*, *13*, 245–286.

Ng, H. T., & Lee, H. B. (1996). Integrating multiple knowledge sources to disambiguate word sense: An exemplar-based approach. In *Proceedings of the 34th Annual Meeting of the Association for Computational Lenguistics*, pp. 40–47 Santa Cruz, CA.

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

Quinlan, J. R., & Rivest, R. L. (1989). Inferring decision trees using the minimum description length principle. *Information and Computation*, *80*, 227–248.

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

Reilly, R. G., & Sharkey, N. E. (Eds.). (1992). *Connectionist Approaches to Natural Language Processing*. Lawrence Erlbaum and Associates, Hilldale, NJ.

Riloff, E. (1993). Automatically constructing a dictionary for information extraction tasks. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pp. 811–816.

Riloff, E. (1996). Automatically generating extraction patterns from untagged text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp. 1044–1049.

Salton, G. (1989). *Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer*. Addison-Wesley.

Seung, H. S., Opper, M., & Sompolinsky, H. (1992). Query by committee.. In *Proceedings of the ACM Workshop on Computational Learning Theory* Pittsburgh, PA.

Soderland, S., Fisher, D., Aseltine, J., & Lehnert, W. (1995). Crystal: Inducing a conceptual dictionary. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp. 1314–1319.

Soderland, S., Fisher, D., Aseltine, J., & Lehnert, W. (1996). Issues in inductive learning of domain-specific text extraction rules. In Wermter, S., Riloff, E., & Scheller, G. (Eds.), *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*, Lecture Notes in Artificial Intelligence, pp. 290–301. Springer.

Srinivasan, A., Muggleton, S., Sternberg, M., & King, R. (1996). Theories for mutagenicity: A study in first-order and feature-based induction. *Artificial Intelligence*, *85*, 277–300.

Thompson, C. A. (1995). Acquisition of a lexicon from semantic representations of sentences. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pp. 335–337 Cambridge, MA.

Weizenbaum, J. (1966). 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.

Wermter, S., Riloff, E., & Scheler, G. (Eds.). (1996). *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*. Springer Verlag, Berlin.

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

Zelle, J. M., & Mooney, R. J. (1996). Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence* Portland, OR.