Relational Learning Techniques for Natural Language Information Extraction

Mary Elaine Califf

Report AI98-276 August 1998

mecalif@ilstu.edu
http://www.cs.utexas.edu/users/mecaliff/

Artificial Intelligence Laboratory
The University of Texas at Austin
Austin, TX 78712

Relational Learning Techniques for Natural Language Information Extraction

by

Mary Elaine Califf, B.A., M.A., M.S.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August 1998

Relational Learning Techniques for Natural Language Information Extraction

Approved by
Dissertation Committee:

To Michael and Jamie.

Acknowledgments

There are many people whom I wish to thank for their support and for their contributions to this research. First, I would like to thank my advisor, Ray Mooney, for all of his help and guidance throughout my time as a graduate student. Working with Ray has been a continuous learning experience. He was always available to help point me in the right direction and I am very grateful for all of his advice and teaching. I also I would also like to thank the other members of my committee, which include Bruce Porter, Risto Miikkulainen, Ben Kuipers, and Ellen Riloff, for their helpful suggestions and comments.

Next, I would like to thank others who have provided help and encouragement, including William Cohen, Robert van de Geijn, and many of my fellow graduate students, among them John Zelle, James Lester, Tara Estlin, and Cindi Thompson. A special acknowledgement is due to Sue Buchele, with whom I shared all of the trials and joys of being a graduate student and a mother. I would also like to thank Ann Hawkins and Shellynne Wucher for their friendship and support.

Finally, I thank my family: my parents, who raised me to believe I really could do anything I truly wanted to do; my husband, who has followed me, supported me, and taken up the slack left by a harried and distracted wife; and my children, who have not always understood why Mommy had to work on the computer all evening, but have tried to be patient and understanding.

This research was supported by a fellowship from AT&T and by the National Science Foundation under grant IRI-9310819.

Mary Elaine Califf

The University of Texas at Austin
August 1998

Relational Learning Techniques for Natural Language Information Extraction

Technical Report AI98-269

Mary Elaine Califf, Ph.D.
The University of Texas at Austin, 1998

Supervisor: Raymond Mooney

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 dissertation presents a novel rule representation specific to natural language and a relational learning system, Rapier, which learns information extraction rules. Rapier takes pairs of documents and filled templates indicating the information to be extracted and learns pattern-matching rules to extract fillers for the slots in the template. The system is tested on several domains, showing its ability to learn rules for different tasks. Rapier's performance is compared to a propositional learning system for information extraction, demonstrating the superiority of relational learning for some information extraction tasks.

Because one difficulty in using machine learning to develop natural language processing systems is the necessity of providing annotated examples to supervised learning systems, this dissertation also describes an attempt to reduce the number of examples Rapier requires by employing a form of *active learning*. Experimental results show that the number of examples required to achieve a given level of performance can be significantly reduced by this method.

Contents

Chapter 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 DARPA'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. An example of the information extraction task which was the focus of MUC-3 and MUC-4 appears in Figures 1.1 and 1.2. The goal was to extract information about Latin American terrorist incidents from news reports.

Information extraction systems seem to be a promising way to deal with certain types of text documents. 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; Aone & 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,

Newswire text

```
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.
```

Figure 1.1: A sample message from the Latin American terrorism domain used in MUC-3 and MUC-4.

1995; Soderland, Fisher, Aseltine, & Lehnert, 1995, 1996; Riloff, 1993, 1996; Kim & Moldovan, 1995; Huffman, 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 classifiers.

Using only a corpus of documents paired with filled templates, 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 which consider only a fixed number of words. 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 (Lavrač & Džeroski, 1994). 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. We show that learning can be used to build useful information extraction rules, and that relational learning is more effective than learning using only simple features and a fixed context.

Experiments using Rapier were performed in three different domains of varying difficulty. In a task of extracting information about computer-related jobs from netnews postings, Rapier performed quite well, achieving recall of 63% and precision of 89%. Rapier was compared to a Naive Bayes-based system which looks only at a fixed window before and after the filler (by default,

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 MINISTER'': ``ENRIQUE
                                     LOPEZ ALBUJAR''
20. HUM TGT: TYPE                  FORMER GOVERNMENT OFFICIAL / FORMER
                                     ACTIVE MILITARY: ``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 1.2: The filled template corresponding to the message shown in Figure 1.1. The slot fillers include both strings found in the documents and other types of values.

4 words), and does not take into account the order of the words, but only their presence or absence. Tests of this Naive Bayes-based system on the jobs task produced much worse results: 32% recall at 14% precision. These results demonstrate the value, in some information extraction tasks, at least, of a richer, relational representation, capable of focusing on a single previous word, if appropriate, or of considering six or more context words when needed, and also capable of taking into account the order of the tokens in both the filler and the context. On an easier task of extracting information from seminar announcements, Rapier also performed well, achieving 92% precision and 71% recall overall, although the Naive Bayes system was more competitive in this domain. Rapier did not perform as well on the third task, which is extracting information about corporate acquisitions from newswire articles. This probably indicates that more syntactic knowledge than Rapier has available is required to successfully handle this type of domain and may also indicate a need for domain-specific semantic information. In all three domains, Rapier is competitive with other state-of-the-art learning systems for information extraction which have been tested on the tasks.

The one difficulty of using machine learning to build systems is the necessity of providing examples to the learning system. In the case of learning information extraction rules, this requires that a human perform the information extraction task on a number of documents. While this is significantly less labor intensive than producing an information extraction system by hand, it is clearly desirable to limit the number of examples required as much as possibly. Therefore, we have experimented with the application of *active learning*, specifically *selective sampling*, to limit the number of examples required to achieve a given level of performance. Selective sampling is a method for allowing a learning system to select examples to be annotated and used for training from a pool of unannotated examples. The method attempts to select the most useful examples in order to reduce the number of annotated example required. Experiments with selective sampling in the computer-related jobs domain show that it is possible to greatly reduce the number of examples required to reach the level of performance achieved with the full set of 270 random examples.

This research has focused on two primary goals. First, we show that learning, and, in particular, relational learning, can be used to build practical information extraction systems. Second, we show that selective sampling can be effectively applied to learning for information extraction to reduce the human effort required to annotate examples for building such systems.

1.1 Organization of Dissertation

The rest of this dissertation is organized as follows. Chapter 2 presents background knowledge on information extraction, relational learning, and the natural language processing tools which the Rapier system can use. Chapter 3 describes the representation used by Rapier and presents the learning algorithm. Chapter 4 discusses the experimental evaluation of Rapier on several domains and presents the results of this evaluation. Chapter 5 describes the application of active learning to Rapier and the extensions required to the algorithm to allow for active learning and discusses the experimental evaluation of Rapier using active learning. Chapter 6 describes related work in the area of learning rules for information extraction. Finally, Chapter 7 suggests ideas for future research directions, and Chapter 8 reviews the ideas and results presented in this dissertation and discusses relevant conclusions.

Chapter 2

Background

The first part of this chapter discusses the information extraction task and characteristics of traditional information extraction systems. The second section discusses relational learning and some the design choices involved in developing a relational rule learning system. That section also describes several previous relational rule learning systems, all of them inductive logic programming systems, including the three systems that directly influenced the development of Rapier. The final section of the chapter briefly describes the natural language processing resources used in this research.

2.1 Information Extraction

Information extraction is a shallow form of natural language understanding useful for certain types of document processing, which has been the focus of ARPA's Message Understanding Conferences (MUC) (Lehnert & Sundheim, 1991; DARPA, 1992, 1993). It is useful in situations where a set of text documents exist containing information which could be more easily used by a human or computer if the information were available in a uniform database format. Thus, an information extraction system is given the set of documents and a template of slots to be filled with information from the document. Information extraction systems locate and in some way identify the specific pieces of data needed from each document.

Two different types of data may be extracted from a document: more commonly, the system is to identify a string taken directly from the document, but in some cases the system selects one from a set of values which are possible fillers for a slot. The latter type of slot-filler may be items like dates, which are most useful in a consistent format, or they may simply be a set of terms to provide consistent values for information which is present in the document, but not necessarily in a consistently useful way. An example of this is in the Latin American terrorism domain used in MUC-3 and MUC-4 (see Figures 1.1 and 1.2), where an incident may be "THREATENED," "ATTEMPTED" or "ACCOMPLISHED."

The data to be extracted may be specified in either of two ways. The system may fill a template with the values from the document, or, in the case where all slots are filled by strings directly from the document, the system may annotate the document directly.

Information extraction can be useful in a variety of domains. The various MUC's have focused on tasks such as the Latin American terrorism domain mentioned above, joint ventures,

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
```

Figure 2.1: A sample job posting from a newsgroup.

microelectronics, and company management changes. Others have used information extraction to track medical patient records (Soderland et al., 1995) and to track company mergers (Huffman, 1996). More recently, researchers have applied information extraction to less formal text genres such as rental ads (Soderland, 1998) and web pages (Freitag, 1998a; Hsu & Dung, 1998; Muslea, Minton, & Knoblock, 1998).

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 databases would be particularly useful as part of a complete NLP system which supported natural language querying of the system. The work on information extraction reported in this dissertation is part of an ongoing project to develop such a system. The initial system handles computer-related jobs only. An example of the information extraction task for the system appears in Figures 2.1 and 2.2.

The architecture of the complete system is shown in figure 2.3. In addition to the information extraction rules learned by Rapier, the system requires a query parser and a semantic lexicon for the query parse. The query parser is being developed using Chill (Zelle & Mooney, 1996), a system which learns parsers from example sentences paired with their parses, and the semantic lexicon is

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 2.2: The filled template corresponding to the message shown in Figure 2.1. All of the slot-fillers are strings taken directly from the document. Not all of the slots are filled, and some have more than one filler.

being produced using WOLFIE (Thompson, 1995; Thompson & Mooney, 1989), 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 (DARPA, 1992, 1993, 1995). 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(Fisher, Soderland, McCarthy, Feng, & Lehnert, 1995; Lehnert, McCarthy, Soderland, Riloff, Cardie, Peterson, Feng, Dolan, & Goldman, 1993).

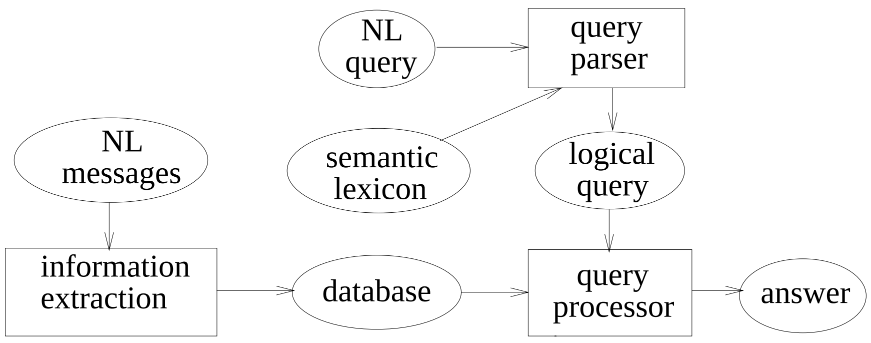

Figure 2.3: Complete System Architecture

2.2 Symbolic Relational Learning

Since much empirical work in natural language processing has employed statistical techniques (Charniak, 1993; Miller, Stallard, Bobrow, & Schwartz, 1996; Smadja, McKeown, & Hatzivassiloglou, 1996; Wermter et al., 1996), 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 et al., 1996; Aone & Bennett, 1995), *exemplar* (*case-based*) methods (Cardie, 1993; Ng & Lee, 1996), and a maximum entropy modeling method (Ratnaparkhi, 1997). 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). Recent research has also demonstrated the usefulness of relational learning in classifying web pages (Slattery & Craven, 1998).

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. This allows the system to make use of the concepts embodied in the background predicates which are relevant to the concept being learned.

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 learn using rich, unbounded representations. Therefore, the following sections discuss some general design issues in developing ILP and other rule learning systems and then describe several ILP systems that influenced Rapier's learning algorithm, including the three which most directly inspired it: Golem, Chillin, and Progol.

2.2.1 General Algorithm Design Issues

One of the design issues in rule learning systems is the overall structure of the algorithm. There are two primary forms for this outer loop: compression and covering. Systems that use compression begin by creating an initial set of highly specific rules, typically one for each example. At each iteration a more general rule is constructed, which replaces the rules it subsumes, thus compressing the rule set. At each iteration, all positive examples are under consideration to some extent, and the metric for evaluating new rules is biased toward greater compression of the rule set. Rule learning ends when no new rules to compress the rule set are found. Systems that use compression include DUCE, a propositional rule learning system using inverse resolution (Muggleton, 1987), CIGOL, an ILP system using inverse resolution (Muggleton & Buntine, 1988), and CHILLIN (Zelle & Mooney, 1994).

Systems that use covering begin with a set of positive examples. Then, as each rule is learned, all positive examples the new rule covers are removed from consideration for the creation of future rules. Rule learning ends when all positive examples have been covered. This is probably the more common way to structure a rule learning system. Examples include FOIL (Quinlan, 1990), GOLEM (Muggleton & Feng, 1992), PROGOL (Muggleton, 1995), CLAUDIEN (De Raedt & Bruynooghe, 1993), and various systems based on FOIL such as FOCL (Pazzani, Brunk, & Silverstein, 1992), MFOIL (Lavrač & Džeroski, 1994), and FOIDL(Mooney & Califf, 1995).

There are trade-offs between these two designs. The primary difference is the trade-off between a more efficient search or a more thorough search. The covering systems tend to be somewhat more efficient, since they do not seek to learn rules for examples that have already been covered. However, their search is less thorough than that of compression systems, since they may not prefer rules which both cover remaining examples and subsume existing rules. Thus, the covering systems may end up with a set of fairly specific rules in cases where a more thorough search might have discovered a more general rule covering the same set of examples.

A second major design decision is the direction of search used to construct individual rules. Systems typically work in one of two directions: bottom-up (specific to general) systems create very specific rules and then generalize those to cover additional positive examples, and top-down (general to specific) systems start with very general rules– typically rules which cover all of the examples, positive and negative, and then specialize those rules, attempting to uncover the negative examples while continuing to cover many of the positive examples. Of the systems above, DUCE, CIGOL, and GOLEM and pure bottom-up systems, while FOIL and the systems based on it are pure top-down systems. CHILLIN and PROGOL both combine bottom-up and top-down methods.

Clearly, the choice of search direction also creates tradeoffs. Top-down systems are often better at finding general rules covering large numbers of examples, since they start with a most general rule and specialize it only enough to avoid the negative examples. Bottom-up systems may create overly-specialized rules that don't perform well on unseen data because they may fail to generalize the initial rules sufficiently. Given a fairly small search space of background relations and constants, top-down search may also be more efficient. However, when the branching factor for a top-down search is very high (as it is when there are many ways to specialize a rule), bottom-up search will usually be more efficient, since it constrains the constants to be considered in the construction of a rule to those in the example(s) that the rule is based on. The systems that combine bottom-up and top-down techniques seek to take advantage of the efficiencies of each.

The following sections briefly describe FOIL and the three systems which most directly influenced RAPIER's algorithm.

```
Initialization
    Definition := null
    Remaining := all positive examples
While Remaining is not empty
        Find a clause, C, that covers some examples in Remaining,
            but no negative examples.
        Remove examples covered by C from Remaining.
        Add C to Definition.
```

Figure 2.4: Basic FOIL Covering Algorithm

2.2.2 FOIL

FOIL is a prototypical example of a top-down ILP algorithm which uses covering for its outer loop. It learns a function-free, first-order, Horn-clause definition of a *target* predicate in terms of itself and other *background* predicates. The input consists of extensional definitions of these predicates as tuples of constants of specified types. For example, input appropriate for learning a definition of list membership would be:

```
member(Elt, Lst): { <a, [a]>, <a, [a, b]>, <b, [a, b]>,
        <a, [a, b, c]>, ...}
components(Lst, Elt, Lst): { <[a], a, []>, <[a, b], a, [b]>,
        <[a, b, c], a, [b, c]> ...}
```

where `Elt` is a type denoting possible elements which includes `a,b,c,` and `d`; `Lst` is a type defined as consisting of flat lists containing up to three of these elements; and `components(A,B,C)` is a background predicate which is true iff `A` is a list whose first element is `B` and whose rest is the list `C` (this must be provided in place of a function for list construction). FOIL also requires negative examples of the target concept, which can be supplied directly or computed using a closed-world assumption. For the example, the closed-world assumption would produce all pairs of the form `<Elt,Lst>` that are not explicitly provided as positive examples (e.g., `<b,[a]>`). Given this input, FOIL learns a program one clause at a time using a greedy covering algorithm that can be summarized as shown in Figure 2.4.

For example, a clause that might be learned for `member` during one iteration of this loop is:

```
member(A,B) :- components(B,A,C).
```

since it covers all positive examples where the element is the first one in the list but does not cover any negatives. A clause that could be learned to cover the remaining examples is:

```
member(A,B) :- components(B,C,D), member(A,D).
```

Together these two clauses constitute a correct program for `member`.

The "find a clause" step is implemented by a general-to-specific hill-climbing search that adds antecedents to the developing clause one at a time. At each step, it evaluates possible literals that might be added and selects one that maximizes an information-gain heuristic. The algorithm

```
Initialize $C$ to $R(V_1, V_2, ..., V_k)$ :-. where $R$ is the target predicate with arity $k$.
Initialize $T$ to contain the positive tuples in positives-to-cover and all the
        negative tuples.
While $T$ contains negative tuples
    Find the best literal $L$ to add to the clause.
    Form a new training set $T'$ containing for each tuple $t$ in $T$ that satisfies $L$,
        all tuples of the form $t \cdot b$ ($t$ and $b$ concatenated) where $b$ is a set of
        bindings for the new variables introduced by $L$ such that the literal is
        satisfied (i.e., matches a tuple in the extensional definition of its
        predicate).
    Replace $T$ by $T'$.
```

Figure 2.5: The "find-a-clause" step in the FOIL algorithm.

maintains a set of tuples that satisfy the current clause and includes bindings for any new variables introduced in the body. The pseudocode is Figure 2.5 summarizes the procedure.

FOIL considers adding literals for all possible variablizations of each predicate as long as type restrictions are satisfied and at least one of the arguments is an existing variable bound by the head or a previous literal in the body. Literals are evaluated based on the number of positive and negative tuples covered, preferring literals that cover many positives and few negatives. Let T_+ denote the number of positive tuples in the set T; then the *informativity* of a clause is defined as:

$$I(T) = -\log_2(T_+/|T|). \tag{2.1}$$

The chosen literal is then the one that maximizes:

$$gain(L) = s \cdot (I(T) - I(T')), \tag{2.2}$$

where s is the number of tuples in T that have extensions in T' (i.e., the number of current positive tuples covered by L). This search for a good literal to add to a clause blows up when the number of background predicates is large, the arity of the predicates is large, or there the number of *theory constants* (constants that may appear in clauses) is very large.

FOIL also includes many additional features such as: heuristics for pruning the space of literals searched, methods for including equality, negation as failure, and useful literals that do not immediately provide gain (*determinate literals*), pre-pruning and post-pruning of clauses to prevent over-fitting, and methods for ensuring that induced programs will terminate. More information on FOIL can be found in (Quinlan, 1990; Quinlan & Cameron-Jones, 1993).

2.2.3 GOLEM

As mentioned above, GOLEM (Muggleton & Feng, 1992) also uses a greedy covering algorithm very similar to FOIL's outer loop. However, the individual clause construction is bottom-up, based on the construction of *least-general generalizations* (LGGs) of more specific clauses (Plotkin, 1970). A clause G *subsumes* a clause C if there is a substitution for the variables in G that make the literals in G a subset of the literals in C. Informally, we could turn C into G by dropping some conditions

```
uncle(john,deb) :-
  sibling(john,ron), sibling(john,dave),
  parent(ron,deb), parent(ron,ben),
  male(john), male(dave), female(deb).
uncle(bill,jay):-
  siblingbill,bruce)
  parent(bruce,jay), parent(bruce,rach),
  male(bill), male(jay).
```

Figure 2.6: Two specific instances of *uncle* relationships

```
uncle(A,B):-
  sibling(A,C), sibling(A,D),
  parent(C,B), parent(C,E), parent(C,F), parent(C,E)
  male(A), male(G), male(H), male(I).
```

Figure 2.7: The LGG of the clauses in Figure 2.6

and changing some constants to variables. If G subsumes C, anything that can be proved from C could also be proved from G, since G imposes fewer conditions. Hence G is said to be more general than C (assuming C does not also subsume G, in which case the clauses must be equivalent except for renaming of variables).

The LGG of clauses C_1 and C_2 is defined as the least general clause that subsumes both C_1 and C_2. An LGG is easily computed by "matching" compatible literals of the clauses; wherever the literals have differing structure, the LGG contains a variable. When identical pairings of differing structures occurs, the same variable is used for the pair in all locations.

For example, consider the clauses in Figure 2.6. These two specific clauses describe the concept `uncle` in the context of some known familial relationships. The rather complex LGG of these clauses is shown in Figure 2.7. Here `A` replaces the pair ⟨`john,bill`⟩, `B` replaces ⟨`deb,jay`⟩, `C` replaces ⟨`ron,bruce`⟩, etc. Note that the result contains four `parent` literals (two of which are duplicates) corresponding to the four ways of matching the pairs of `parent` literals from the original clauses. Similarly, there are four literals for `male`. In the worst case, the result of an LGG operation may contain n^2 literals for two input clauses of length n. The example LGG contains no `female` literal since the second clause does not contain a compatible literal. Straightforward simplification of the result by removing redundant literals yields the clause in Figure 2.8. This is one of the two clauses defining the general concept, `uncle/2`.

The construction of the LGG of two clauses is in some sense "context free." The resulting generalization is determined strictly on the form of the input clauses, there is no consideration of potential background knowledge. In order to take background knowledge into account GOLEM produces candidate clauses by considering *Relative* LGGs (RLGGS) of positive examples with respect to the background knowledge. The idea is to start with the assumption that any and all background information might be relevant to determining that a particular instance is a positive

```
uncle(A,B):-
  sibling(A,C) parent(C,B), male(A).
```

Figure 2.8: The result of simplifying the clause from Figure 2.7 by removing redundant clauses

```
Let $Pairs$ = random sampling of pairs of positive examples
Let $RLggs = \{C : \langle e, e' \rangle \in Pairs$ and $C = RLGG(e, e')$ and $C$ consistent$\}$
Let S be set of the pair $e, e'$ with best cover RLgg in $RLggs$
Do
    Let $Examples$ be a random sampling of positive examples
    Let $RLggs = \{C$: $e' \in Examples$ and $C = RLGG(S \bigcup e'))$ and $C$ consistent$\}$
    Find $e'$ = which produces greatest cover in $RLggs$
    Let $S = S \bigcup e'$
    Let $Examples = Examples - cover(RLGG(S))$
While increasing-cover
```

Figure 2.9: GOLEM's clause construction algorithm

example. Thus, each positive example is represented by a clause of the form: `E :-` ⟨`every ground fact`⟩ where ⟨`every ground fact`⟩ is a conjunction of all true ground literals that can be derived from the background relations. In the case of `member/2`, this would include facts such as `components([1],1,[])`, `components([1,2],1,[2])`, `components([2],2,[])`, etc. An RLGG of two examples is simply the LGG of the examples' representative clauses. The LGG process serves to generalize away the irrelevant conditions.

One difficulty of this approach is that interesting background relations will give rise to an infinite number of ground facts. For example, there can be no finite set of facts that completely describes the `components/3` relation, since lists may be indefinitely long. GOLEM builds initial representative clauses for examples by considering a finite subset corresponding to the facts that can be derived from the background predicates through a fixed number of binary resolutions. Figure 2.9 shows GOLEM's clause construction algorithm.

2.2.4 CHILLIN

CHILLIN (Zelle & Mooney, 1994) is an example of an ILP algorithm that uses compression for its outer loop. It combines elements of both top-down and bottom-up induction techniques including a mechanism for demand-driven predicate invention. The basic compaction algorithm appears in Figure 2.10.

CHILLIN starts with a most specific definition (the set of positive examples) and introduces generalizations which make the definition more compact (as measured by a CIGOL-like size metric (Muggleton & Buntine, 1988)). The search for more general definitions is carried out in a hill-climbing fashion. 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. To determine which clauses in the current theory a new clause should replace, CHILLIN uses a notion of

```
DEF := {E :- true | E ∈ Positives}
Repeat
  PAIRS := a sampling of pairs of clauses from DEF
  GENS := {G | G = build_gen($C_i$,$C_j$,DEF,Positives,Negatives) for $\langle C_i, C_j \rangle$ ∈ PAIRS}
  G := Clause in GENS yielding most compaction
  DEF := (DEF−(Clauses subsumed by G)) ∪ G
Until no further compaction
```

Figure 2.10: CHILLIN's compaction algorithm

empirical subsumption. If a clause A covers all of the examples covered by clause B along with one or more additional examples, then A empirically subsumes B.

The `build_gen` algorithm attempts to construct a clause which empirically subsumes some clauses of `DEF` without covering any of the negative examples. The first step is to construct the LGG of the input clauses. If the LGG does not cover any negative examples, no further refinement is necessary. If the clause is too general, an attempt is made to refine it using a FOIL-like mechanism which adds literals derivable either from background or previously invented predicates. If the resulting clause is still too general, it is passed to a routine which invents a new predicate to discriminate the positive examples from the negatives which are still covered.

2.2.5 PROGOL

PROGOL (Muggleton, 1995) also combines bottom-up and top-down search. Like FOIL and GOLEM, PROGOL uses a covering algorithm for its outer loop. As in the propositional rule learner AQ (Michalski, 1983), individual clause construction begins by selecting a random seed example. Using mode declarations provided for both the background predicates and the predicate being learned, PROGOL constructs a most specific clause for that random seed example, called the *bottom* clause. 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 the bottom clause, PROGOL employs an A*-like search through the set of clauses containing up to k literals from the bottom 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 Natural Language Processing Resources

One issue in any kind of higher level language processing, such as information extraction, is what kinds of lower level processing are possible and helpful. A number of information extraction systems use parsers and lexicons (usually at least partially domain specific) developed in conjunction with the information extraction system. However, dependence upon such parsers or lexicons could limit the usefulness of a system intended to learn information extraction rules in order to be easily retargetable to new tasks. Such a system would benefit most from using language processing resources which are more general and likely to be of help in a variety of domains.

Two current types of language processing resources which are general enough to apply across multiple domains are part-of-speech taggers and domain-independent lexicons.

2.3.1 Part-of-speech tagger

A part-of-speech (POS) tagger provides basic syntactic information by taking sentences as input and labeling 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, since it doesn't identify phrases or the relationships between parts of the sentence. However, taggers are typically faster and more robust than full parsers, particularly in the face of ungrammatical text such as would be commonly found in newgroup postings and email messages, and to a lesser extent in newswire articles.

The particular tagger used in this research is Eric Brill's tagger as trained on a Wall Street Journal corpus (Brill, 1994, 1995). This tagger uses 36 different tags excluding punctuation, so it does make fairly specific distinctions in some cases: for example, it identifies six different verb forms plus modals; it distinguishes wh-determiners, pronouns, possessive pronouns, and adverbs from other determiners, pronouns, and adverbs; and it also distinguishes between "to" and other prepositions. Appendix C contains a list of the part-of-speech tags used, excluding punctuation. Brill's tagger is also quite accurate on the domain for which it was trained, achieving an accuracy of 96.6% on the Penn Treebank Wall Street Journal corpus (Brill, 1995). Naturally, its accuracy on other domains is significantly lower, but it does have the advantage of being trainable. If a large amount of hand-tagged text happens to exist for a given domain, the tagger can be trained from scratch. In the absence of such text, several methods exist for tuning the tagger for a specific domain. First, Brill provides methods to incorporate new words and bigrams from a new corpus into the resources used by the tagger. Second, the lexicon, which is used by the tagger to determine what parts of speech a word can be and what its most common part of speech is, can be modified to include new words from the new domain, or to reflect the actual distribution of parts of speech in the new domain. Adding the most frequent novel words to the lexicon seems to be the most effective way of improving tagging quality. Finally, the rules used by the tagger are quite comprehensible, and new rules can be added by hand to improve the tagging quality. This is a much more difficult and time-consuming process, since it requires a good understanding of what the tagger is doing wrong and how to fix it. Using the first two methods for tuning the tagger can typically be done in two to four hours, depending on the size of the domain.

2.3.2 Lexicon

Lexicons, particularly those with semantic hierarchies, can provide semantic class information. In this research, the domain-independent lexicon used was WordNet (Miller, Beckwith, Fellbaum, Gross, & Miller, 1993; Fellbaum, 1998), a lexical database of over 50,000 words which contains a semantic hierarchy in the form of *hypernym* links.

The individual items in WordNet are *synsets* which represent a single meaning, often one which is shared by more than one word. For instance, one sense of “man” is shared by “humanity”, ”humankind”, “world”, and “mankind”. Each word typically has several synsets to which it belongs. For example, “man” is a member of ten noun synsets and 2 verb synsets. Within each part of speech are the various senses of a word: the mappings from word to synset are ordered by frequency. The synset mentioned above is the first sense of “humanity”, the second sense of “world”, and the third sense of “man”. The synsets in WordNet are related by links of various types, including antonyms, synonyms, meronyms, holonyms, hyponyms, and hypernyms. The semantic hierarchy is implemented in the hyponym and hypernym links. Hyponym links point to semantic subclasses while hypernym links point to semantic superclasses. The semantic hierarchy described by these links is a forest rather than a single tree, even for each part of speech. For example, the top of the hierarchy for the first sense of “man” (adult male) is “entity”, while the top of the hierarchy for the third sense of “man” (humanity) is “grouping”. Thus, two arbitrary synsets in the same part of speech do not necessarily share a semantic class.

Chapter 3

The Rapier System

Rapier learns rules for information extraction from training examples consisting of documents paired with filled templates such as those in Figures 2.1 and 2.2. The next two sections describe Rapier's rule representation and learning algorithm.

3.1 Rule Representation

Rapier's rule representation uses Eliza-like patterns (Weizenbaum, 1966) that can make use of limited syntactic and semantic information. 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 elements. Rapier makes use of two types of pattern elements: *pattern items* and *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 (a limited form of Kleene closure), each of which must match the list's constraints. Rapier uses three kinds of constraints on pattern elements: constraints on the words the element can match, on the part-of-speech tags assigned to the words the element can match, and on the semantic class of the words the element can match. The constraints are disjunctive lists of one or more words, tags, or semantic classes and document items must match one of those words, tags, or classes to fulfill the constraint.

Figure 3.1 shows an example of a rule that shows the various types of pattern elements and constraints. This is a rule constructed by Rapier for extracting the transaction amount from a newswire concerning a corporate acquisition. This rule extracts the value "undisclosed" from phrases such as "sold to the bank for an undisclosed amount" or "paid Honeywell an undisclosed price". The pre-filler pattern consists of two pattern elements. The first is an item with a part-of-speech constraining the matching word to be tagged as a noun or a proper noun. The second is a list of maximum length two with no constraints. The filler pattern is a single item constrained to be the word "undisclosed" with a POS tag labeling it an adjective. The post-filler pattern is also a single pattern item with a semantic constraint of "price".

In using these patterns to extract information, we apply all of the rules for a given slot to a document and take all of the extracted strings to be slot-fillers, eliminating duplicates. Rules may

Pre-filler Pattern:	Filler Pattern:	Post-filler Pattern:
1) syntactic: {nn,nnp}	1) word: undisclosed	1) semantic: price
2) list: length 2	syntactic: jj	

Figure 3.1: A rule for extracting the transaction amount from a newswire concerning a corporate acquisition. "nn" and "nnp" are the part of speech tags for noun and proper noun, respectively; "jj" is the part of speech tag for an adjective.

also apply more than once. In many cases, multiple slot fillers are possible, and the system seldom proposes multiple fillers for slots where only one filler should occur.

3.2 Learning Algorithm

Rapier, as noted above, is inspired by ILP methods, particularly by Golem, Chillin, and Progol. It is compression-based and primarily consists of a specific to general (bottom-up) search. The choice of a bottom-up approach was made for two reasons. The first reason is the very large branching factor of the search space, particularly in finding word and semantic constraints. Learning systems that operate on natural language typically must have some mechanism for handling the search imposed by the large vocabulary of any significant amount of text (or speech). Many systems handle this problem by imposing limits on the vocabulary considered–using only the n most frequent words, or considering only words that appear at least k times in the training corpus (Yang & Pederson, 1997). While this type of limitation may be effective, using a bottom-up approach reduces the consideration of constants in the creation of any rule to those appearing in the example(s) from which the rule is being generalized, thus limiting the search without imposing artificial hard limits on the constants to be considered.

The second reason for selecting a bottom-up approach is that we decided to prefer overly specific rules to overly general ones. In information extraction, as well as other natural language processing task, there is typically a trade-off between high precision (avoiding false positives) and high recall (identifying most of the true positives). For the task of building a database of jobs which partially motivated this work, we wished to emphasize precision. After all, the information in such a database could be found by performing a keyword search on the original documents, giving maximal recall (given that we extract only strings taken directly from the document), but relatively low precision. A bottom-up approach will tend to produce specific rules, which also tend to be precise rules.

Given the choice of a bottom-up approach, the compression outer loop is a good fit. A bottom-up approach has a strong tendency toward producing specific, precise rules. Using compression for the outer loop may partially counteract this tendency with its tendency toward a more thorough search for general rules. So, like Chillin (Zelle & Mooney, 1994), Rapier begins with a most specific definition and then attempts to compact that definition by replacing rules with more general rules. Since in Rapier's rule representation rules for the different slots are independent of one another, the system actually creates the most specific definition and then compacts it separately for each slot in the template.

3.2.1 Creation of the Initial Rulebase

Thus, the first step for each slot is the creation of the most-specific rulebase for that slot. For each document, rules are created for each occurrence of each slot-filler in that document's template. The filler pattern of the rule is a list of pattern items, one for each word or symbol in the filler, with a word constraint which is the word or symbol and a tag constraint which is the part-of-speech tag from the document. The semantic class is left unconstrained because a single word often has multiple possible semantic classes because of the homonymy and polysemy of language. If semantic constraints were immediately created, RAPIER would have to either use a disjunction of all of the possible classes at the lowest level of generality (in the case of WordNet– the synsets that the word for which the item is created belongs to) or choose a semantic class. The first choice is somewhat problematic because the resulting constraint is quite likely to be too general to be of much use. The second choice is the best, if and only if the correct semantic class for the word in context is known, a difficult problem in and of itself. Selecting the most frequent choice from WordNet might work for some cases, but certainly not in all cases, and there is the issue of domain specificity. The most frequent meaning of a word in all contexts may not be the most frequent meaning of that word in the particular domain in question. And, of course, even within a single domain words will have multiple meanings so even determining the most frequent meaning of a word in a specific domain may often be a wrong choice. RAPIER avoids the issue altogether by waiting to create semantic constraints until generalization. Thus, it implicitly allows the disjunction of classes, selecting a specific class only when the item is generalized against one containing a different word. By postponing the choice of a semantic class until there are multiple items required to fit the semantic constraint, RAPIER narrows the number of possible choices for the semantic class to classes that cover two or more words. Details concerning the creation of semantic constraints are discussed in Section 3.2.4.

The creation of the pre-filler and post-filler patterns proceeds in the same fashion as the filler pattern. The pre-filler pattern is a list of of pattern items for all of the document tokens preceding the filler in the document, and the post-filler pattern consists of items for all of the document tokens after the filler. Thus, each rule covers only the one slot-filler for which it was created, and it provides full information (within the limits of the rule representation) about one occurrence of that filler and its context.

3.2.2 Compression of the Rulebase

Once it has created the most-specific definition for a slot, RAPIER attempts to compact the rulebase by replacing specific rules with more general ones. The more general rules are created by taking several random pairs of rules from the rulebase, finding generalizations of those rule pairs, and, if the generalizations result in one or more acceptable rules, selecting the best rule to add to the rulebase. When the new rule is added to the rulebase, RAPIER removes the old rules which it empirically subsumes, i.e. the ones which cover a subset of the examples covered by the new rule.

As indicated in Chapter 2, inspiration for the decision to create new rules by generalizing pairs of rules came from GOLEM (Muggleton & Feng, 1992). However, RAPIER differs from GOLEM significantly in its use of this basic concept. First, GOLEM always selects random pairs of *examples*, and RAPIER selects random pairs of *rules*. When the rules selected by RAPIER are the most-specific rules covering a single example, there is no real difference between these two, since an example in

Inductive Logic Programming is essentially a most-specific rule. However, RAPIER may select rules that resulted from an earlier generalization and generalize them further. This points to a second difference between the two learning algorithms: the difference in the way they repeat generalization to achieve rules as general as possible. While GOLEM takes the rule resulting from generalizing a pair of examples and attempts to generalize it further to cover new, randomly selected examples, RAPIER simply adds the rule to the rulebase where it may be later selected for generalization with another rule. These differences stem primarily from the differing approaches of the two algorithms at the outer level. Since GOLEM takes a covering approach, it must fully generalize a given rule initially, since the examples the rule covers will be removed from further consideration. RAPIER's compression approach leads it to do less generalization at each loop.

The most unique aspect of RAPIER's learning algorithm is the way in which it actually creates a new rule from a random pair of rules. The straightforward method of generalizing two rules together would be find the least general generalization (LGG) of the two pre-filler patterns and use that as the pre-filler pattern of the new rule, make the filler pattern of the new rule be the LGG of the two filler patterns, and then do the same for the post-filler pattern. There are, however, two serious problems with this obvious approach.

The first problems is the expense of computing the LGGs of the pre-filler and post-filler patterns. These patterns may be very long, and the pre-filler or post-filler patterns of two rules may be of different lengths. As is further discussed in Section 3.2.4, generalizing patterns of different lengths is computationally expensive because each individual pattern element in the shorter pattern may be generalized against one or more elements of the longer pattern, and it is not known ahead of time how elements should be combined to produce the LGG. Thus, the computation of the LGG of the pre-filler and post-filler patterns in their entirety may be prohibitively computationally expensive.

The second problem is not a matter of computational complexity, but rather a problem caused by the power of the rule representation. Because RAPIER's rule representation allows for unlimited disjunction, the LGG of two constraints is always their union. When the two constraints differ, the resulting disjunct may be the desirable generalization, but often a better generalization results from simply removing the constraint instead of creating the disjunction. Therefore, when generalizing pattern elements, rather than simply taking the LGG of the constraints, it is useful to consider multiple generalizations if the constraints on the pattern elements differ, and this is the approach the RAPIER takes – considering for each pair of differing constraints the generalization created by dropping the constraint as well as the generalization which is the union of the constraints. However, considering multiple generalizations of each pair of pattern elements greatly increases the computational complexity of generalizing pairs of lengthy patterns such as the most-specific pre-fillers and post-fillers tend to be.

Because of these issues, RAPIER does not use a pure bottom-up approach. Instead, like PROGOL, it combines the bottom-up approach with a top-down component, using a specific rule to constrain a top-down search for an acceptable general rule. However, instead of using a single seed or using some type of user-provided information (such as PROGOL's modes), RAPIER uses a pair of rules to constrain the search, more like CHILLIN does. This approach, along with the difference in representation, makes RAPIER's top-down search very different from PROGOL's.

RAPIER's rule generalization method operates on the principle that the relevant information

```
For each slot, S in the template being learned
   SlotRules = most specific rules for S from example documents
   while compression has failed fewer than CompressLim times
      initialize RuleList to be an empty priority queue of length k
      randomly select M pairs of rules from SlotRules
      find the set L of generalizations of the fillers of each rule pair
      for each pattern P in L
         create a rule NewRule with filler P and empty pre- and
            post-fillers
         evaluate NewRule and add NewRule to RuleList
      let n = 0
      loop
         increment n
         for each rule, CurRule, in RuleList
            NewRuleList = SpecializePreFiller (CurRule, n)
            evaluate each rule in NewRuleList and add it to RuleList
         for each rule, CurRule, in RuleList
            NewRuleList = SpecializePostFiller (CurRule, n)
            evaluate each rule in NewRuleList and add it to RuleList
      until best rule in RuleList produces only valid fillers or
            the value of the best rule in RuleList has failed to
            improve over the last LimNoImprovements iterations
      if best rule in RuleList covers no more than an allowable
            percentage of spurious fillers
         add it to SlotRules and remove empirically subsumed rules
```

Figure 3.2: Rapier Algorithm for Inducing Information Extraction Rules

for extracting a slot-filler will be close to that filler in the document. Therefore, Rapier begins by generalizing the two filler patterns and creates rules with the resulting generalized filler patterns and empty pre-filler and post-filler patterns. It then specializes those rules by adding pattern elements to the pre-filler and post-filler patterns, working outward from the filler. The elements to be added to the patterns are created by generalizing the appropriate portions of the pre-fillers or post-fillers of the pair of rules from which the new rule is generalized. Working in this way takes advantage of the locality of language, but does not preclude the possibility of using pattern elements that are fairly distant from the filler. Further details of the specialization process appear below in Section 3.2.5.

Figure 3.2 shows Rapier's basic algorithm. *RuleList* is a priority queue of length k which maintains the list of rules still under consideration, where k is a parameter of the algorithm. The priority of the rule is its value according to Rapier's heuristic metric for determining the quality of a rule (see Section 3.2.3). Rapier's search is basically a beam-search: a breadth-first search keeping only the best k items at each pass. However, the search does differ somewhat from a standard beam-search in that the nodes (or rules) are not fully expanded at each pass (since at each iteration the specialization algorithms only consider pattern elements out to a distance of n from the filler as will be further described in Section 3.2.5), and because of this the old rules are only thrown out when they fall off the end of the priority queue.

3.2.3 Rule Evaluation Metric

One difficulty in designing the Rapier algorithm was in determining an appropriate heuristic metric for evaluating the rules being learned. The first issue is the measurement of negative examples. Clearly, in a task like information extraction there are a very large number of possible negative examples – strings which should not be extracted – a number large enough to make explicit enumeration of the negative examples difficult, at best. Another issue is the question of precisely which substrings constitute appropriate negative examples: should all of the strings of any length be considered negative examples, or only those strings with lengths similar to the positive examples for a given slot. To avoid these problems, Rapier does not enumerate the negative examples, but uses a notion of implicit negatives instead (Zelle, Thompson, Califf, & Mooney, 1995). First, Rapier makes the assumption that all of the strings which should be extracted for each slot are specified, so that any strings which a rule extracts that are not specified in the template are assumed to be spurious extractions and, therefore, negative examples. Whenever a rule is evaluated, it is applied to each document in the training set. Any fillers that match the fillers for the slot in the training templates are considered positive examples; all other extracted fillers are considered negative examples covered by the rule.

Given a method for determining the negative as well as the positive examples covered by the rule, a rule evaluation metric can be devised. Because Rapier does not use a simple search technique such as hill-climbing, it cannot use a metric like information gain (Quinlan, 1990) which measures how much each proposed new rule improves upon the current rule in order to pick the new rule with the greatest improvement. Rather, each rule needs an inherent value which can be compared with all other rules. One such value is the informativity of each rule (see Equation 2.1). However, while informativity measures the degree to which a rule separates positive and negative examples (in this case, identifies valid fillers but not spurious fillers), it makes no distinction between simple and complex rules. The problem with this is that, given two rules which cover the same number of positives and no negatives but different levels of complexity (one with two constraints and one with twenty constraints), we would expect the simpler rule to generalize better to new examples, so we would want that rule to be preferred. Many machine learning algorithms encode such a preference; all top-down hill-climbing algorithms which stop when a rule covering no negatives is found have this preference for simple rather than complex rules. Since Rapier's search does not encode such a preference, but can, because of its consideration of multiple ways of generalizing constraints, produce many rules of widely varying complexities at any step of generalization or specialization, the evaluation metric for the rules needs to encode a bias against complex rules. Finally, we want the evaluation metric to be biased in favor of rules which cover larger number of positive examples.

One evaluation metric which clearly seems to fit the requirements is the *minimum description length* (MDL) (Rissanen, 1978; 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 listing the positive examples not yet covered plus correction for negative examples covered). MDL seems an excellent choice for a system like Rapier. It provides a principled way to measure both the accuracy and the complexity of the rules covered. It also fits as easily with a compression outer loop as with a covering outer loop.

In actually using MDL, the primary issue is how to measure the size of the theory and the corrections. The principled way to do this is to compute the minimal number of bits required to transmit the theory and the corrections. Initial experiments using MDL with this method of measuring the size of the theory and corrections yielded somewhat disappointing results. The recall was in the expected range, but the precision was much lower than expected for the task.

One way of adjusting the performance of the MDL metric is to modify the method of measurement of the theory length and the corrections. Therefore, an alternate size metric was developed, one which is deliberately designed to reflect the specificity of the various aspects of the rules. The size of the rule is computed by a simple heuristic as follows: each pattern item counts 2; each pattern list counts 3; each disjunct in a word constraint counts 2; and each disjunct in a POS tag constraint or semantic constraint counts 1. Because of the compression style loop, there is no need to measure positive examples, since they are guaranteed to be covered by the theory. Each covered negative example counts a fixed size (several different values for this size were tried). This alternative measurement heuristic did slightly increase the precision over the principled metric, but still left precision much lower than expected. Although MDL has a strong theoretical base, it does not necessarily prefer accurate theories and has previously been shown to perform poorly on certain tasks, although adding penalties and using an alternative encoding improved performance (Quinlan, 1994, 1995). Like other model selection methods, MDL merely provides a bias for preferring theories, and it seems not to be the correct bias for this task.

The performance of the MDL metric indicates that it is giving too much weight to the complexity of the rules and insufficient weight to the rules' accuracy. One way to solve this problem is to use a metric which computes accuracy and complexity in different terms which could then be weighted appropriately to allow each to influence the evaluation. Putting together all of the requirements for a desirable evaluation metric described above, an appropriate metric might be to take the informativity of the rule and weight that by the size of the rule divided by the number of positives covered. This is the metric that Rapier uses. The informativity is computed using the Laplace estimate of the probabilities. The size of a rule is computed in the same way as the second MDL metric described above and then divided by 100 to bring the heuristic size estimate into a range which allows the informativity and the rule size to influence each other, with neither value being overwhelmed by the other. Thus the evaluation metric is computed as:

$$ruleVal = -\log_2(\frac{p+1}{p+n+2}) + \frac{ruleSize}{p}$$

where p is the number of correct fillers extracted by the rule and n is the number of spurious fillers the rule extracts.

Noise handling

Rapier does allow coverage of some spurious fillers. The primary reason for this is that human annotators make errors, especially errors of omission. If Rapier rejects a rule covering a large number of positives because it extracts a few negative examples, it can be prevented from learning useful patterns by the failure of a human annotator to notice even a single filler that fits that pattern which should, in fact, be extracted. If Rapier's specialization ends due to failure to improve on the best rule for too many iterations and the best rule still extracts spurious examples, the best rule is used if it meets the criteria:

$$\frac{p-n}{p+n} > noiseParam$$

where p is the number of valid fillers extracted by the rule and n is the number of spurious fillers extracted. This equation is taken from RIPPER (Cohen, 1995), which uses it for pruning rules measuring p and n using a hold-out set. Because RAPIER is usually learning from a relatively small number of examples, it does not use a hold-out set or internal cross-validation in its evaluation of rules which cover spurious fillers, but uses a much higher default value of *noiseParam* (Cohen uses a default of 0.5; RAPIER's default value is 0.9).

Note that RAPIER's noise handling does not involve pruning, as noise handling often does. Pruning is appropriate for top-down approaches, because in noise handling the goal is to avoid creating rules that are too specialized and over-fit the data and, in pure top-down systems, the only way to generalize a too-specific rule is some sort of pruning. Since RAPIER is a primarily bottom-up compression-based system, it can depend on subsequent iterations of the compression algorithm to further generalize any rules that may be too specific. The noise handling mechanism need only allow the acceptance of noisy rules when the "best" rule, according to the rule evaluation metric, covers negative examples.

3.2.4 Computing the Generalizations of Two Patterns

The description of RAPIER's algorithm thus far has given the overall structure of the algorithm and has described its search pattern but has left vague three important steps: the generalization of a pair of fillers and the two specialization phases: specialization of the pre-filler pattern and specialization of the post-filler pattern. Crucial to an understanding of these three steps is the understanding of how RAPIER generalizes pairs of patterns. Therefore, this section describes how this is done, starting by describing the generalization of constraints, then the generalization of pattern elements, and finally the generalization of patterns.

Constraint Generalization

The generalization of a pair of word or tag constraints is straightforward. The only issue involved in this generalization is that simply taking the LGG of the constraints will always produce a disjunction, and it is preferable to consider both the disjunction and the more general option of simply dropping the constraint. Throughout the following discussion, an empty constraint is used to describe the result of dropping a constraint.

In three cases, RAPIER does simply take the LGG as the only appropriate generalization. Clearly, if the two constraints are identical, then the new constraint is simply the same as the original constraints. If either constraint is empty (the word or tag is unconstrained), then the generalized constraint is also empty. If either constraint is a superset of the other, the new constraint will be the superset. The reason for only creating the disjunction in this case is that the disjunction which is the superset must have been one of two generalizations created, so the result of dropping the constraint either is still under consideration elsewhere or has been rejected in favor of the disjunction.

In all other cases, two alternative generalizations are created: one is the union of the two constraints and one is the empty constraint. Thus, the results of generalizing {*nn, nnp*} and {*adj*} are {*nn, nnp, adj*} and the empty tag constraint.

The generalization of semantic constraints is more complex for two reasons. First, as mentioned above, because a word can be a member of many semantic classes, RAPIER does not create semantic constraints in the initial rules, but instead waits until generalization so that a single semantic class covering at least two different words is chosen, making it more likely that the semantic class is a useful generalization. The second cause for the greater complexity is the use of a semantic hierarchy. Rather than simply creating disjunctions of classes, the system seeks to find a superclass covering all of classes in the initial constraints. Given these issues, the generalization of semantic constraints proceeds as follows:

- If the two constraints are non-empty and identical, the new constraint is the same as the original constraints.
- If both constraints are empty and either pattern element has an empty word constraint or a word constraint containing more than one word, the new constraint is empty. In this case, the constraints in the element with the empty word constraint or the word constraint with multiple words must be the result of a previous generalization. Since the semantic class is unconstrained, search for an appropriate semantic constraint must have already failed.
- If both semantic constraints are empty and the pattern elements' word constraints each consist of a single word and the two word constraints differ, RAPIER attempts to create a semantic constraint. The system searches the semantic hierarchy for a class which covers both of the words in the word constraints. The goal is to find the least general class which covers a meaning for each of the words. In WordNet, this means finding the synsets for the two words and searching, following the hypernym links, for the synset which is closest to a synset for each word. If the two words are "man"and "world", the semantic class will be the synset shared by "humanity", "mankind", "world", "humankind", and "man", which is the second synset for "world" and the third synset for "man". The words "man" and "woman" would result in the semantic class "person", while the words "man" and "rock" would result in "entity". If no match is found, the new semantic constraint will be empty. RAPIER avoids creating a semantic constraint when the two word constraints are identical for the same reason that it does not create semantic constraints for the initial rules: having a only a single word leaves the choice of a semantic class insufficiently constrained. This will not prevent it from creating a semantic class later, because the word constraint in the rule will still be a single word.
- If one of the semantic constraints is empty and the other is not, there are two cases:
    1. If the pattern element with the empty semantic constraint also has an empty word constraint or a word constraint with multiple words, the generalized constraint will be empty.
    2. Otherwise, the system starts with the semantic class in the semantic constraint and climbs the semantic hierarchy as necessary to find a semantic class which covers the word in the first element's word constraint. That semantic class will be the new semantic constraint.

Elements to be generalized

Element A	Element B
word: man	word: woman
syntactic: nnp	syntactic: nn
semantic:	semantic:

Resulting Generalizations

word:	word: {man, woman}
syntactic: {nn, nnp}	syntactic: {nn, nnp}
semantic: person	semantic: person
word:	word: {man, woman}
syntactic:	syntactic:
semantic: person	semantic: person

Figure 3.3: An example of the generalization of two pattern elements. The words "man" and "woman" form two possible generalizations: their disjunction and dropping the word constraint. The tags "nn" (noun) and "nnp" (proper noun) also have two possible generalizations. Thus, there are a total of four generalizations of the two elements.

- Finally, if both semantic constraints are non-empty, the system searches for the lowest class in the semantic hierarchy which is a superclass of both constraints, and makes that the new semantic constraint. If no matching superclass is found, the new constraint is empty. Thus, for semantic classes, RAPIER does not actually make use of disjunction in the current implementation.

It should be noted that the implementation of semantic constraints and their generalization is very closely tied to WordNet (Miller et al., 1993) since that is the semantic hierarchy used in this research. However, the code has been carefully modularized in order to make the process of substituting an alternative source for semantic information or modifying the generalization method to allow for disjunctions of classes relatively easy.

Generalizing Pattern Elements

Given the rules for generalizing constraints, the generalization of a pair of pattern elements is fairly simple. First, the generalizations of the word, tag and semantic constraints of the two pattern elements are computed as described above. From that set of generalizations, RAPIER computes all combinations of a word constraint, a tag constraint, and the semantic constraint and creates a pattern element with each combination. See Figure 3.3 for an example of this combination. If both of the original pattern elements are pattern items, the new elements are pattern items as well. Otherwise, the new elements are pattern lists. The length of these new pattern lists is the maximum length of the original pattern lists (or the length of the pattern list in the case where a pattern item and a pattern list are being generalized).

Patterns to be Generalized

Pattern A	Pattern B
1) word: ate	1) word: hit
syntactic: vb	syntactic: vb
2) word: the	2) word: the
syntactic: dt	syntactic: dt
3) word: pasta	3) word: ball
syntactic: nn	syntactic: nn

Resulting Generalizations

1) word: {ate, hit}	1) word:
syntactic: vb	syntactic: vb
2) word: the	2) word: the
syntactic: dt	syntactic: dt
3) word: {pasta, ball}	3) word: {pasta, ball}
syntactic: nn	syntactic: nn

1) word: {ate, hit}	1) word:
syntactic: vb	syntactic: vb
2) word: the	2) word: the
syntactic: dt	syntactic: dt
3) word:	3) word:
syntactic: nn	syntactic: nn

Figure 3.4: Generalization of a pair of patterns of equal length. For simplicity, the semantic constraints are not shown, since they never have more than one generalization.

Generalizing Patterns

Generalizing a pair of patterns of equal length is also quite straightforward. Rapier pairs up the pattern elements from first to last in the patterns and computes the generalizations of each pair. It then creates all of the patterns made by combining the generalizations of the pairs of elements in order. Figure 3.4 shows an example.

Generalizing pairs of patterns that differ in length is more complex, and the problem of combinatorial explosion is greater. Suppose we have two patterns: one five elements long and the other three elements long. We need to determine how to group the elements to be generalized. If we assume that each element of the shorter pattern must match at least one element of the longer pattern, and that each element of the longer pattern will match exactly one element of the shorter pattern, we have a total of three ways to match each element of the shorter pattern to elements of the longer pattern, and a total of six ways to match up the elements of the two patterns. Figures 3.5 and 3.6 demonstrate the problem. As the patterns grow longer and the difference is length grows larger, the problem becomes more severe.

In order to limit this problem somewhat, before creating all of the possible generalizations, Rapier searches for any exact matches of two elements of the patterns being generalized, making the assumption that if an element from one of the patterns exactly matches an element of the other pattern then those two elements should be paired and the problem broken into matching

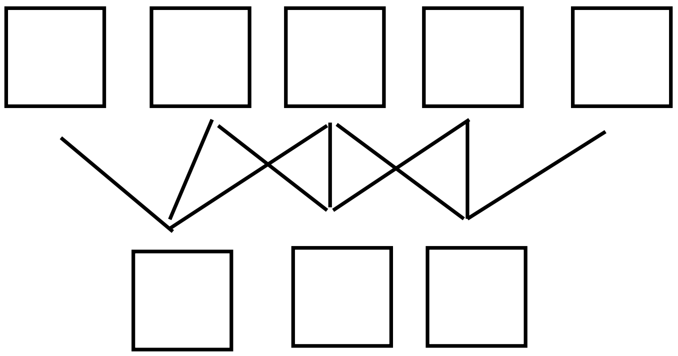

Figure 3.5: Two patterns of differing lengths. The lines show the various elements that may be grouped together for generalization.

the segments of the patterns to either side of these matching elements. However, the search for matching elements is confined by the first assumption of matching above: that each element of the shorter pattern should be generalized with at least one element of the longer pattern. Thus, if the shorter pattern, A, has three elements and the longer, B, has five, the first element of A is compared to elements 1 to 3 of B, element 2 of A to elements 2-4 of B, and element 3 of A to elements 3-5 of B. If any matches are found, they can greatly limit the number of generalizations that need to be computed. Figure 3.7 shows an example where finding an exact match is very helpful. Since the third element of the longer pattern exactly matches the second element of the shorter, those elements are paired, leaving only one way to combine the remaining pattern elements.

Any exact matches that are found break up the patterns into segments which still must be generalized. Each pair of segments can be treated as a pair of patterns that need to be generalized, so if any corresponding pattern segments are of equal length, they are handled just like a pair of patterns of the same length as described above. Otherwise, we have patterns of uneven length that must be generalized.

There are three special cases of different length patterns. First, the shorter pattern may have 0 elements. In this case, the pattern elements in the longer pattern are generalized into a set of pattern lists, one pattern list for each alternative generalization of the constraints of the pattern elements. Each of the resulting pattern lists must be able to match as many document tokens as the elements in the longer pattern, so the length of the pattern lists is the sum of the lengths of the elements of the longer pattern, with pattern items naturally having a length of one. Figure 3.8 demonstrates this case.

The second special case is when the shorter pattern has a single element. This is similar to the previous case, with each generalization again being a single pattern list, with constraints generalized from the pattern elements of both patterns. In this case the length of the pattern lists is the greater of the length of the pattern element from the shorter pattern or the sum of the lengths of the elements of the longer pattern. The length of the shorter pattern must be considered in case it is a list of length greater than the length of the longer pattern. A example of this case appears in Figure 3.9.

The third special case is when the two patterns are long or very different in length. In this case, the number of generalizations becomes very large, so RAPIER simply creates a single pattern

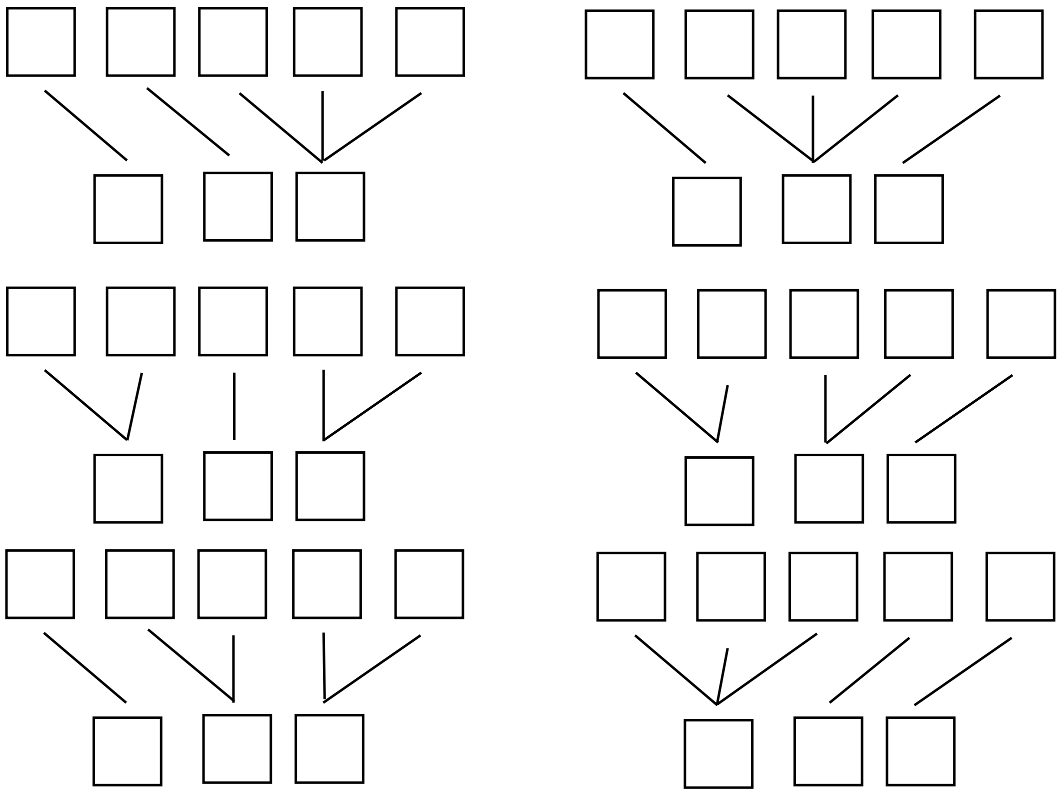

Figure 3.6: The six possible ways the elements of the patterns in Figure 3.5 may be matched up for generalization.

list with no constraints and a length equal to the longer of the two patterns (measuring sums of lengths of elements). This case happens primarily with slot fillers of very disparate length, where there is unlikely to be a useful generalization, and any useful rule is likely to make use of the context rather than the structure of the actual slot filler.

When none of the special cases holds, Rapier must create the full set of generalizations as described above. Rapier creates the set of generalizations of the patterns by first creating the generalizations of each of the elements of the shorter pattern against each possible set of elements from the longer pattern using the assumptions mentioned above: each element from the shorter pattern must correspond to at least one element from the longer pattern and each element of the longer pattern corresponds to exactly one element of the shorter pattern for each grouping. Once all of the possible generalizations of elements are computed, the generalizations of the patterns are created by combining the possible generalizations of the elements in all possible combinations which include each element of each pattern exactly once in order. In the case of a pattern with 3 elements and one with 5, as in Figure 3.5, the generalizations of the pattern would be all of the combinations of generalizations of elements corresponding to the six possible groupings of pattern elements that appear in Figure 3.6. Since each generalization of elements may produce up to four generalizations, there will typically be far more than six generalizations computed.

In the case where exact matches were found, one step remains after the various resulting pattern segment pairs are generalized. The generalizations of the patterns are computed by creating all possible combinations of the generalizations of the pattern segment pairs. In the case depicted

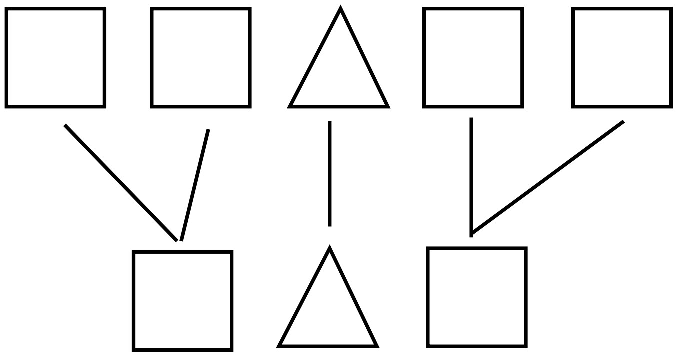

Figure 3.7: The grouping that results from finding an exact match between element 3 of the longer pattern and element 2 of the shorter pattern from Figure 3.5. Once the identical elements are paired, the remainder can be grouped in only one way.

Pattern to be Generalized
1) word: bank
   syntactic: nn
2) word: vault
   syntactic: nn

Resulting Generalizations

1) list: length 2 word: {bank, vault} syntactic: nn	1) list: length 2 word: syntactic: nn

Figure 3.8: Generalization of two pattern items matched with no pattern elements from the other pattern.

in Figure 3.7, this results in up to sixteen generalizations, as each generalization for the first pattern segment is combined with the middle element of the patterns and each generalization of the final pattern segment.

3.2.5 The Specialization Phase

The final piece of the learning algorithm is the specialization phase, indicated by calls to SpecializePreFiller and SpecializePostFiller in Figure 3.2. These functions take two parameters, the rule to be specialized and an integer n which indicates how many elements of the pre-filler or post-filler patterns of the original rule pair are to be used for this specialization. As n increments, the specialization uses more context, working outward from the slot-filler. In order to carry out the specialization phase, each rule maintains information about the two rules from which it was created, which are referred to as the base rules: pointers to the two base rules, how much of the pre-filler pattern from each base rule has been incorporated into the current rule, and how much of the post-filler pattern from each base rule has been incorporated into the current rule. The two specialization functions return a list of rules which have been specialized by adding to the rule

Patterns to be Generalized

Pattern A	Pattern B
1) word: bank	1) list: length 3
syntactic: nn	word:
2) word: vault	syntactic: nnp
syntactic: nn	

Resulting Generalizations

1) list: length 3	1) list: length 3
word:	word:
syntactic: {nn,nnp}	syntactic:

Figure 3.9: Generalization of two pattern items matched with one pattern element from the other pattern. Because Pattern B is a pattern list of length 3, the resulting generalizations must also have a length of 3.

generalizations of the appropriate portions of the pre-fillers or post-fillers of the base rules.

One issue arises in these functions. If the system simply considers adding one element from each pattern at each step away from the filler, it may miss some useful generalizations since the lengths of the two patterns being generalized would always be the same. For example, assume we have two rules for required years of experience created from the phrases “6 years experience required” and “4 years experience is required.” Once the fillers were generalized, the algorithm would need to specialize the resulting rule(s) to identify the number as years of experience and as required rather than desired. The first two iterations would create items for “years” and “experience,” and the third iteration would match up “is” and “required.” It would be helpful if a fourth iteration could match up the two occurrences of “required,” creating a list from “is.” In order to allow this to happen, the specialization functions do not only consider the result of adding one element from each pattern; they also consider the results of adding an element to the first pattern, but not the second, and adding an element to the second pattern but not the first.

Pseudocode for SpecializePreFiller appears in Figure 3.10 and that for SpecializePostFiller appears in Figure 3.11. In order to allow pattern lists to be created where appropriate, the functions generalize three pairs of pattern segments. The patterns to be generalized are determined by first determining how much of the pre-filler (post-filler) of each of the original pair of rules the current rule already incorporates. Using the pre-filler case as an example, if the current rule has an empty pre-filler, the three patterns to be generalized are: 1) the last n elements of the pre-filler of *BaseRule1* and the last $n-1$ elements of the pre-filler of *BaseRule2*, 2) the last $n-1$ elements of the pre-filler of *BaseRule1* and the last n elements of the pre-filler of *BaseRule2*, and 3) the last n elements of the pre-filler of each of the base rules. If the current rule has already been specialized with a portion of the pre-filler, then whatever elements it already incorporates will not be used, but the pattern of the pre-filler to be used will start at the same place, so that n is not the number of elements to be generalized, but rather specifies the portion of the pre-filler which can be considered at that iteration.

The post-filler case is analogous to the pre-filler case except that the portion of the pattern

```
**SpecializePreFiller**(*CurRule*,*n*)
Let *BaseRule1* and *BaseRule2* be the two rules from which *CurRule* was created
Let *CurPreFiller* be the pre-filler pattern of *CurRule*
Let *PreFiller1* be the pre-filler pattern of *BaseRule1*
Let *PreFiller2* be the pre-filler pattern of *BaseRule2*
Let *PatternLen1* be the length of *PreFiller1*
Let *PatternLen2* be the length of *PreFiller2*
Let *FirstUsed1* be the first element of *PreFiller1* that has been used in *CurRule*
Let *FirstUsed2* be the first element of *PreFiller2* that has been used in *CurRule*
GenSet1 = Generalizations of elements $(PatternLen1 + 1 - n)$ to FirstUsed1 of
        PreFiller1 with elements $(PatternLen2 + 1 - (n - 1))$ to FirstUsed2 of
        PreFiller2
GenSet2 = Generalizations of elements $(PatternLen1 + 1 - (n - 1))$ to
        FirstUsed1 of PreFiller1 with elements $(PatternLen2 + 1 - n)$ to
        FirstUsed2 of PreFiller2
GenSet3 = Generalizations of elements $(PatternLen1 + 1 - n)$ to FirstUsed1 of
        PreFiller1 with elements $(PatternLen2 + 1 - n)$ to FirstUsed2 of
        PreFiller2
GenSet = $GenSet1 \cup GenSet2 \cup GenSet3$
NewRuleSet = empty set
For each PatternSegment in GenSet
    NewPreFiller = PatternSegment concatenate CurPreFiller
    Create NewRule from CurRule with pre-filler NewPreFiller
    Add NewRule to NewRuleSet
Return NewRuleSet
```

Figure 3.10: Rapier Algorithm for Specializing the Pre-Filler of a Rule

to considered is that at the beginning, since the algorithm works outward from the filler.

3.2.6 Example

As an example of the entire process of creating a new rule, consider generalizing the rules based on the phrases "located in Atlanta, Georgia." and "offices in Kansas City, Missouri." These phrases are sufficient to demonstrate the process, though rules in practice would be much longer. The initial, specific rules created from these phrases for the city slot for a job template would be

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

and

```
SpecializePostFiller(CurRule,n)
Let BaseRule1 and BaseRule2 be the two rules from which CurRule was created
Let CurPostFiller be the post-filler pattern of CurRule
Let PostFiller1 be the post-filler pattern of BaseRule1
Let PostFiller2 be the post-filler pattern of BaseRule2
Let PatternLen1 be the length of PostFiller1
Let PatternLen2 be the length of PostFiller2
Let LastUsed1 be the last element of PostFiller1 that has been used in CurRule
Let LastUsed2 be the last element of PostFiller2 that has been used in CurRule
GenSet1 = Generalizations of elements LastUsed1 to n of PostFiller1 with
        elements LastUsed2 to $(n - 1)$ of PostFiller2
GenSet2 = Generalizations of elements LastUsed1 to $(n - 1)$ of PostFiller1 with
        elements LastUsed2 to n of PostFiller2
GenSet3 = Generalizations of elements LastUsed1 to n of PostFiller1 with
        elements LastUsed2 to n of PostFiller2
GenSet = GenSet1 $\cup$ GenSet2 $\cup$ GenSet3
NewRuleSet = empty set
For each PatternSegment in GenSet
    NewPostFiller = CurPostFiller concatenate PatternSegment
    Create NewRule from CurRule with post-filler NewPostFiller
    Add NewRule to NewRuleSet
Return NewRuleSet
```

Figure 3.11: Rapier Algorithm for Specializing the Post-Filler of a Rule

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

For the purposes of this example, we assume that there is a semantic class for states, but not one for cities. For simplicity, we assume the beam-width is 2. 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. At the first iteration of specialization, the algorithm considers the first pattern item to either side of the filler. This results in:

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

and

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

The items produced from the "in"'s and the commas are identical and, therefore, unchanged. Alternative, but less useful rules, will also be produced with lists in place of the items in the pre-filler and post-filler patterns because of specializations produced by generalizing the element from each pattern with no elements from the other pattern. Continuing the specialization with the two alternatives above only, the algorithm moves on to look at the second to last elements in the pre-filler pattern. This generalization of these elements produce six possible specializations for each of the rules in the current beam:

list: length 1	list: length 1	word: {located, offices}
word: located	word: offices	tag: {vbn, nns}
tag: vbn	tag: nns	
word:	word:	word: {located, offices}
tag: {vbn, nns}	tag:	tag:

None of these specializations is likely to improve the rule, and specialization proceeds to the second elements of the post-fillers. Again, the two pattern lists will be created, one for the pattern item from each pattern. Then the two pattern items will be generalized. Since we assume that the 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	1) list: max length: 2	1) word: ,
tag: in	tag: nnp	tag: ,
		2) tag: nnp
		semantic: state

Chapter 4

Experimental Evaluation

Our purpose in developing RAPIER is to show that machine learning can be used to develop useful information extraction systems and that relational learning is particularly suitable for this task. To this purpose, we have tested RAPIER on three different information extraction tasks in fairly different domains. We compare RAPIER's performance on these tasks with a non-relational learning system – a Naive Bayes-based learner which uses a fixed window of contextual information. We also compare RAPIER to two other relational learning systems which recently have been developed by other researchers. Additionally in these experiments, we examine the usefulness of the various types of constraints which RAPIER can employ by running ablation tests.

4.1 Domain and Task Descriptions

The first domain consists of a set of 300 computer-related job postings from the Usenet newsgroup `austin.jobs`. The information extraction task is to identify the types of information that would be useful is creating a searchable database of such jobs, with fields like message-id and the posting date which are useful for maintaining the database, and then fields that describe the job itself, such as the job title, the company, the recruiter, the location, the salary, the languages and platforms used, and required years of experience and degrees. Some of these slots can take only one value, but for most of the slots a job posting can contain more than one appropriate slot-filler. There are a total of 17 different slots for this task.

The various slots in this task differ considerably in their difficulty for humans. Some, such as the id and posting date, are trivial. Other fairly obvious slots are salary, the various location slots, and required and desired degrees and years of experience. The position title is usually clear, but not invariably: for instance, one has to decide whether "Unix Programmer needed" should produce "Programmer" or "Unix Programmer." Companies and recruiters are fairly easy to recognize, but not always easy to distinguish. There are four slots that provide information specific to computer-related jobs: language, platform (which includes operating systems, machine names, and brands), application, and area (which serves to cover all other interesting items – broad areas of computer science such as artificial intelligence, natural language processing, or networking; APIs; application frameworks such as MFC; and subject areas such as the World Wide Web. These slots require a lot of domain specific knowledge. It can be very difficult to determine from context whether an unfamiliar term belongs in one of these slots. The area slot is particularly difficult.

The second domain is a set of 485 announcements of seminars collected by Dayne Freitag at Carnegie Mellon University (Freitag, 1997). The task is to extract four pieces of information: start time, end time, location, and speaker. Each of these slots takes only one value, though that value may come in multiple forms (e.g. "2pm" and "2:00 pm" for the start time). Therefore, a slot is considered filled correctly as long as any one of the possible correct strings is extracted.

The third domain is a set of 600 Reuters newswire articles concerning corporate acquisitions also collected at Carnegie Mellon (Freitag, 1998c).The information extraction task identifies 13 slots. Nine of the slots identify the purchaser, seller, and entity acquired by full name, abbreviation, and code. The other four slots are the business and location of the entity being acquired, the dollar amount of the transaction, and the status of the acquisition.

Examples of documents and the corresponding filled templates for each domain can be found in Appendix A.

4.2 Data Pre-processing

Although Rapier does not require full syntactic analysis, some pre-processing is required to segment the text into tokens and sentences appropriate for use by the part-of-speech tagger and then to tag the text. For the experiments in this chapter, the documents were segmented using a simple Perl script which separates words from punctuation, recognizing common abbreviations (using a list of common abbreviations to avoid sentence breaks based on periods used for abbreviation), and breaks the document into sentences at end-of-sentence punctuation and blank lines. The results of this pre-processing were tagged using Brill's part-of-speech tagger as trained on the Wall Street Journal corpus and tuned for each domain using the techniques described in Section 2.3.1. Initial results with Rapier indicated that part-of-speech tagging without domain-specific tuning was not helpful, and inspection of the tagging results indicated that they were inaccurate, so for each domain bigram and word lists were created, and the most common words in each corpus were added to the lexicon with their possible parts of speech. This tuning took about two hours per domain.

4.3 Ablations

The Rapier system is designed to be able to use semantic classes and part of speech tags in its rules. However, it is important to consider the utility of each kind of information. Therefore, for the experiments presented here, three versions of Rapier were used: the full system using words, part of speech tags, and semantic classes from WordNet; an ablation using only words (labeled Rapier-w in tables); and an ablation using word and part-of-speech tags (labeled Rapier-wt). Comparing these versions of Rapier provides information about how useful the various types of information are.

4.4 Systems Compared Against

The first system that we compare to is a Naive Bayes-based information extraction system developed at Carnegie Mellon (Freitag, 1998b). This classifier uses word occurrence as its only feature. It looks at the filler and fixed window around the filler and estimates the probability that a particular segment of text is a slot-filler based on the estimated probability that each word in the text segment is in a filler for the slot and the probability that each word in the window preceding the segment is in the left context window for the slot and that each word in the window following the segment is in the right context window for the slot. One issue which this type of system has to cope with is that slot-fillers are not necessarily uniform in length. Thus, the system considers multiple text segments beginning at each word, with lengths from the shortest filler seen for the slot to the longest filler seen for the slot. By default, the Naive Bayes system considers four words to either side of the potential slot-filler, a context which is often sufficient for a human to determine whether a string is, in fact, a slot-filler. However, the system's propositional nature may keep this context from being sufficient. The system recognizes whether words are likely to appear in a filler or its context, but cannot represent the structure of a filler or the greater probability of a string being a filler if two words appear in conjunction rather than separately. The experiments with Naive Bayes show what is possible with a simple set of features and demonstrate the superiority of relational learning on slots where the simple propositional representation is insufficient.

Very recently, two other learning systems have been developed with goals very similar to those of RAPIER. These are both relational learning systems which do not depend on syntactic analysis. Their representations and algorithms, however, differ significantly from each other and from RAPIER.

The first of these systems is SRV (Freitag, 1998b, 1998a, 1998c). SRV is a top-down covering learner, with an algorithm similar to FOIL. It uses four pre-determined predicates which allow it to express information about the length of a fragment, the position of a particular token, the relative positions of two tokens, and the result of feature value tests for tokens using user-defined features. Thus, the language is quite expressive and has the ability to take advantage of orthographic, syntactic, and semantic information if such information is provided in the user-defined features. For example, in the experiments reported on here, SRV was supplied with features such as capitalization, whether a token was a number, whether a token was a single digit number, and features describing the length of the token, among others. Many of these features are quite specific and particularly useful for the seminar announcement domain. SRV also does rule accuracy estimation, using threefold internal cross-validation to provide confidences for its rules.

The second system is WHISK (Soderland, 1998), which is designed to handle all types of extraction problems, from very structured text to the semi-structured text common to web documents and netnews postings to free text such as news articles. Like Rapier, WHISK uses a pattern-matching representation: a restricted form of regular expressions. It can make use of user-provided semantic classes and of information from syntactic analysis, but it does not require either type of information. The learning algorithm is a covering algorithm, and rule creation begins by selection of a seed example. However, WHISK creates the rule top-down, only restricting the choice of terms to be added to a rule to those appearing in the seed example. WHISK differs from SRV and RAPIER in its rule representation and operation in two significant ways. First, a single rule may extract items for more than one slot. Second, rather than operating on documents, WHISK operates on *instances*, which are typically sentences or similarly sized units. Thus, it somewhat limits the amount of context that a given rule may consider.

One interesting feature of WHISK is its use of active learning. WHISK is designed to use a form of selective sampling. It uses neither confidences nor voting, but rather random selects a pool of instances from each of three classes of the unlabeled instances: 1) those covered by an existing rule, to increase support for the reliable rules and provide counter-examples to imprecise rules; 2) those that are near misses of an existing rule, providing likely useful instances to increase recall; 3) and those not covered by any rule, to help ensure the completeness of the rule base.

Since SRV and WHISK are also relational learners, we do not expect to show that RAPIER is significantly better than either. Rather, by examining their performance along with RAPIER's, we show that RAPIER is competitive with recent systems, and we provide further evidence for the usefulness of relational learning for information extraction.

4.5 Experimental Methodology

For all of the experiments in this research, we use averages of ten trials and report results on separate test data. For the computer-related jobs domain, we used ten-fold cross-validation and also trained on subsets of the training data at various sizes in order to produce learning curves. For the seminar announcement and corporate acquisitions domains, we use the ten splits used by Dayne Freitag in order to permit comparison with his results. These were 10 random splits of the datasets in half, training on half of the data and testing on the other half.

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 fixed 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.

$$precision = \frac{\#\ of\ correct\ fillers\ extracted}{\#\ of\ fillers\ extracted}$$

$$recall = \frac{\#\ of\ correct\ fillers\ extracted}{\#\ of\ fillers\ in\ correct\ templates}$$

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 in some respect. Lower recall indicates that the system is failing to find correct fillers: that its rules are too specific in some respect. When multiple fillers are possible for a slot, each filler is counted separately. Since these two numbers best describe the performance of the system we report them for all experiments in this chapter. In addition to precision and recall, we report two other measures for some experiments. The MUC conferences have introduced an F-measure (DARPA, 1992), combining precision and recall in order to provide a single number measurement for information extraction systems. The F-measure is computed as:

$$F = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

with β used to weight the measure to prefer better precision or better recall. Since the F-measure provides a useful tool for examining the relative performance of systems when one has better precision and the other better recall, we report that number where it is useful, using β equal to 1. Finally, in order to facilitate comparison to other systems, we in some cases report coverage, which is the percentage of the slots which are supposed to be filled which had some filler predicted for them. This figure is not as useful as recall, but it is the figure reported for Dayne Freitag's systems on the seminar announcement and corporate acquisition domains, so we include it in addition to recall in order to allow some comparison to be made.

Where tests of statistical significance are appropriate, they were conducted using a two-tailed paired t-test.

All of the experiments in this chapter were conducted using Rapier set to its default parameters.

NumPairs = 5. The number of pairs randomly selected for generalization at each iteration of the compression algorithm.

BeamWidth = 6. The width of the beam for the beam search.

MaxCompressFails = 3. The number of times in a row which the attempt to create a new, more general rule is allowed to fail before compression of the rules for a given slot ends.

MaxExtendFails = 3. The number of times in a row which specialization loop is allowed to iterate without improving the best rule in the beam.

MinCoverage = 3. The number of positive examples which a rule is required to cover.

4.6 Job Postings Results

In the computer-related job postings domain, we conducted experiments with four systems: the three versions of Rapier with different information supplied and the Naive Bayes-based information extraction system described above. The Naive Bayes system provides a baseline indicating what can be learned relying on word occurrence in the filler and a fixed context. The default width of the window the system looks at is four words to either side of the filler, but preliminary tests showed that raising this value to seven words improved performance on the job postings domain, so that value was used for this experiment. The system also has a threshold parameter which was set to maximize recall.

Figure 4.1 shows the learning curve for precision of the four systems and Figure 4.2 shows their recall learning curve. Since precision and recall do not present the same picture, Figure 4.3 presents the F-measure learning curve.

Clearly, the Naive Bayes system does not perform well on this task, although it has been shown to be fairly competitive in other domains, as will be seen below. Actually, there are some slots on which Naive Bayes does perform well, but it does quite poorly on many, especially those for which multiple fillers are common. It should be noted that the very low setting of the Naive Bayes system's threshold accounts for its very low precision. However, recall is also considerably lower than any of the Rapier systems, and raising the threshold to raise precision will simultaneously lower recall, which is already quite low, significantly lower than any version of Rapier. These results indicate that this is a domain where relational learning helps and a simpler representation such as the Naive Bayes system employs is insufficient to produce a useful system.

In looking at the results for the various Rapier systems, there are several points of interest. First, precision is quite high, over 89% for words only and word with POS tags and just over 70%

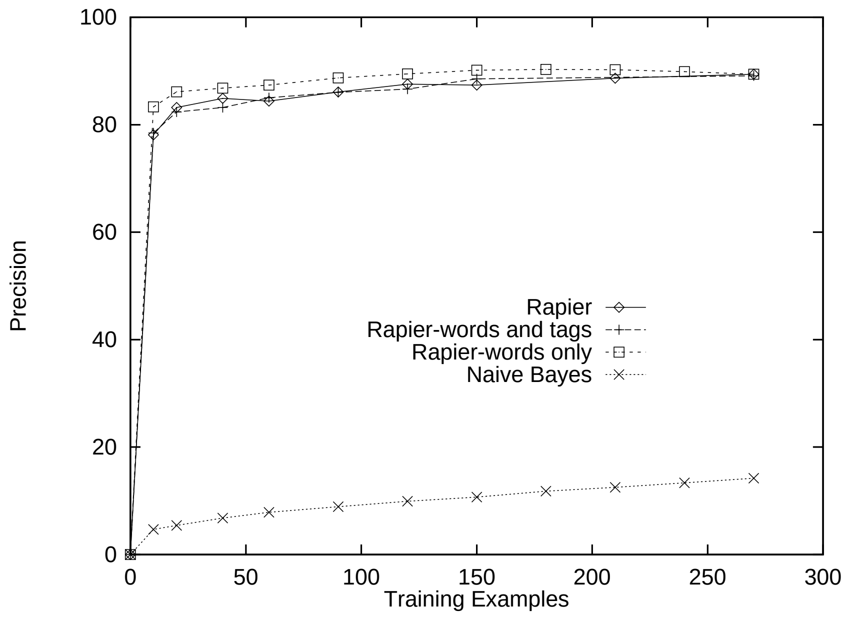

Figure 4.1: Precision on computer-related job postings

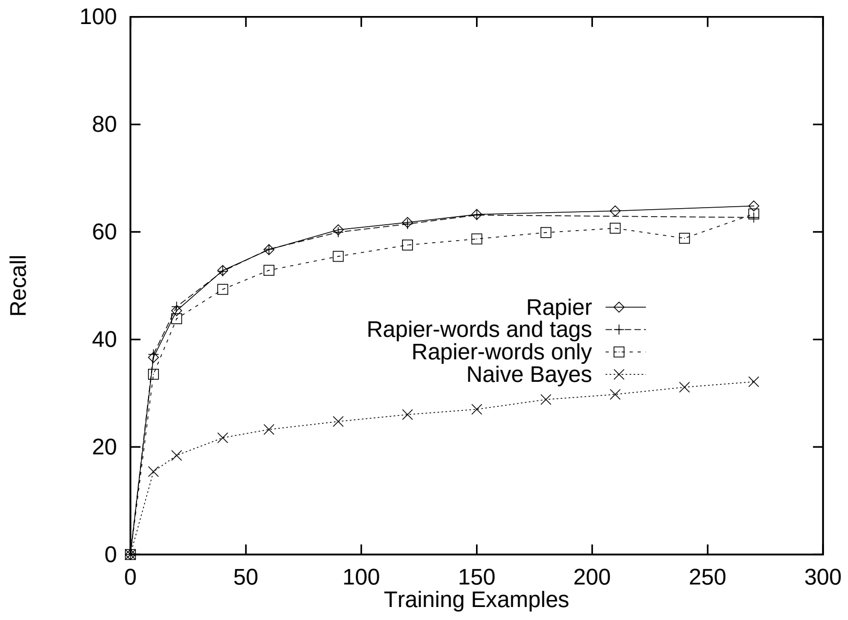

Figure 4.2: Recall on computer-related job postings

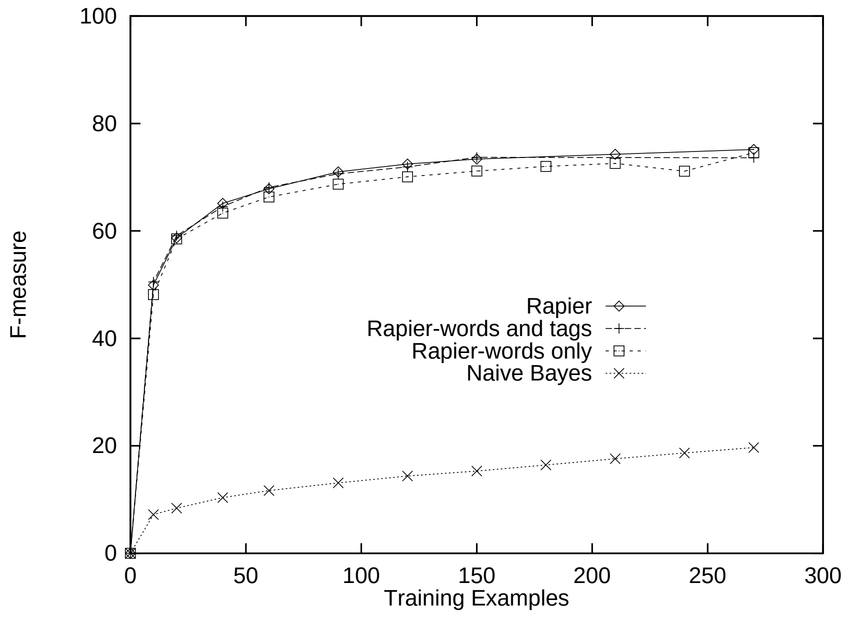

Figure 4.3: F-measure on computer-related job postings

for the full system. This fact is not at all surprising, since the bias of the algorithm is very much toward fairly specific rules, a natural result of a primarily bottom-up approach. High precision is important for a task such as creating a jobs database. Although high recall is desirable (we want as much of the correct information in the database as possible), it is even more important that the information in the database be correct.

A second point of interest is that the learning curve is quite steep. The RAPIER algorithm is apparently quite effective at making maximal use of a small number of examples. The learning curve flattens out quite a bit as the number of examples increases. This is probably due at least in part to the fact that this domain includes quite a few documents that are very similar to one another, so adding more examples may not significantly increase the amount of useful information available to the learning system. This problem may be usefully addressed with active learning (as discussed in Chapter 5). Note, however, that recall is still rising, though slowly, at 270 examples.

In looking at the performance of the three versions of RAPIER, an obvious conclusion is that word constraints provide most of the power of the rules. This should not be particularly surprising, since the information provided by part-of-speech tags and semantic classes are, in some sense, present in the words themselves. What the addition of POS tag constraints and semantic class constraints should do is to provide generalizations that may be representable with word constraints alone, but are simpler and more easily cover a complete class of words without having to see every possible word in the class (as a disjunction of words must in order to represent the same concept). The addition of tag constraints does improve performance at lower number of examples. The recall of the version with tag constraints is significantly better at least at the 0.05 level for each point on the training curve up to 120 examples. Apparently, by 270 examples, the word constraints are capable of representing the concepts provided by the POS tags. The differences between words-only and words with POS tags on 270 examples are not statistically significant. WordNet's semantic classes provide no significant performance increase over words and POS tags only.

These results are very encouraging. They are similar to the scores of the better systems at MUC-6 on the template elements task (the one most similar to this information extraction task) (DARPA, 1995). They are also sufficient to construct a useful, if imperfect, database of computer-related jobs.

One other learning system, WHISK (Soderland, 1998), has been applied to this data set. In a single trial using 200 documents for training and the remaining 100 for testing, WHISK achieved 76% precision and 40% recall. At 180 examples, RAPIER with words only achieves 90% precision and 60% recall. Of course, since WHISK was tested on a single trial, it is important not to over-emphasize this result. However, RAPIER's worst single trial at 180 examples gave 85% precision and 54% recall. It is not clear why WHISK performs worse on this task. One possibility is its restriction of context to a single sentence. RAPIER learns rules which cross sentence boundaries in this domain, and these may be needed for some slots. Both relational learning systems clearly perform much better on this task than the Naive Bayes-based system does.

Besides simply looking at the summary numbers, it is useful to consider the accuracy of the learning systems on each of the different slots. As indicated above, the difficulty of the slots varies for a human annotator, and it may be useful to see how the difficulty varies for the computer system. Accordingly, Table 4.1 shows the results for the various slots with 270 training examples.

The easiest fields for the computer are also some of the easiest for human beings. All of the systems perform well on the id slot, and the RAPIER systems perform very well on the posting date, where Naive Bayes achieves 100% recall, although with low precision. The location slots, company, recruiter, years of experience, and degrees required are all somewhat more difficult for the RAPIER systems, but RAPIER's performance on those is also quite good. The somewhat lower recall on company and recruiter is probably due to the difficulty of distinguishing between the two fields. Unlike humans, RAPIER performs rather poorly on the title slot. This is a slot where memorization is not very helpful and context is inconsistent. The other slot on which RAPIER performs quite poorly is the area slot. The difficulties of this slot for humans was discussed above, and it is not surprising that RAPIER performs poorly here. Besides the two easy slots, Naive Bayes achieves reasonable recall on the three location slots and on company and recruiter. On all of the other slots it does extremely poorly. Much of this is probably due to the system's inability to consider the order of tokens and to the fixed context window.

4.7 Seminar Announcement Results

For the seminar announcements domain, we ran experiments with the three versions of RAPIER, and we report those results along with results reported on this data using the same 10 data splits with the Naive Bayes system mentioned above and with SRV (Freitag, 1998b). The dataset consists of 485 documents, and this was randomly split approximately in half for each of the 10 runs. Thus training and testing sets were approximately 240 examples each. The results for the other systems are reported by individual slots only, and the figures reported are for "accuracy" (which is measured exactly the same way as precision) and coverage. However, a lower bound on recall can be computed by multiplying the precision by the coverage. We also report results for WHISK. These results are from a single trial, trained on 285 of the documents and tested on the remaining 200. As above, because the results are based on a single trial, one should not be overly confident in comparisons

System	id		title		salary	
	Prec	Rec	Prec	Rec	Prec	Rec
Rapier	98.0	97.0	67.0	29.0	89.2	54.2
Rap-wt	98.0	96.7	66.1	31.1	76.9	46.7
Rap-w	97.7	97.0	67.6	30.7	80.0	56.1
NaiBay	86.8	96.3	13.8	18.5	2.1	9.3
	company		recruiter		state	
	Prec	Rec	Prec	Rec	Prec	Rec
Rapier	76.0	64.8	87.7	56.0	93.5	87.1
Rap-wt	77.0	64.7	90.7	59.0	95.2	85.3
Rap-w	75.3	65.1	88.5	60.2	96.5	82.8
NaiBay	9.3	51.1	18.9	49.4	29.2	59.5
	city		country		language	
	Prec	Rec	Prec	Rec	Prec	Rec
Rapier	97.4	84.3	92.2	94.2	95.3	71.6
Rap-wt	97.5	86.1	91.5	93.5	95.5	70.2
Rap-w	97.0	86.1	93.4	92.0	94.0	73.7
NaiBay	33.3	55.4	14.4	62.3	13.9	9.8
	platform		application		area	
	Prec	Rec	Prec	Rec	Prec	Rec
Rapier	92.2	59.7	87.5	57.4	66.6	31.1
Rap-wt	90.1	57.9	85.8	57.7	69.0	31.4
Rap-w	89.7	53.5	88.7	53.8	68.2	29.5
NaiBay	6.5	6.6	4.0	7.9	7.4	5.6
	req yrs exp		des yrs exp		req degree	
	Prec	Rec	Prec	Rec	Prec	Rec
Rapier	80.7	57.5	94.6	81.4	88.0	75.9
Rap-wt	82.8	62.7	90.6	67.4	86.8	79.3
Rap-w	80.9	58.2	94.3	76.7	87.8	74.1
NaiBay	4.1	10.5	1.0	14.0	0.8	8.6
	des degree		post date		total	
	Prec	Rec	Prec	Rec	Prec	Rec
Rapier	86.7	61.9	99.3	99.7	89.4	64.8
Rap-wt	100	61.9	99.0	99.3	89.2	64.6
Rap-w	92.3	57.1	99.3	100	89.4	64.0
NaiBay	0	0	21.5	100	14.2	32.1

Table 4.1: Results by slot for the computer-related job postings task

System	stime			etime			loc		
	Prec	Rec	Cov	Prec	Rec	Cov	Prec	Rec	Cov
RAPIER	93.9	92.9	98.9	95.8	94.6	96.3	91.0	60.5	66.3
RAP-WT	96.5	95.3	98.8	94.9	94.4	97.0	91.0	61.5	67.4
RAP-W	96.5	95.9	99.3	96.8	96.6	97.3	90.0	54.8	60.8
NAIBAY	98.2	$\geq$ 98.2	100	96.1	$\geq$ 95.72	99.6	59.6	$\geq$ 58.8	98.7
SRV	98.6	$\geq$ 98.4	99.8	94.1	$\geq$ 92.6	98.4	75.9	$\geq$ 70.1	92.3
WHISK	89	89	-	96	74	-	93	59	-

System	speaker		
	Prec	Rec	Cov
RAPIER	80.9	39.4	46.7
RAP-WT	79.0	40.0	48.6
RAP-W	76.9	29.1	39.8
NAIBAY	36.1	$\geq$ 25.6	70.8
SRV	60.4	$\geq$ 58.3	96.6
WHISK	71	15	-

Table 4.2: Results for seminar announcements task

between the systems based on them. Table 4.2 shows results for the six systems on the four slots for the seminar announcement task. The figures for recall for Naive Bayes and SRV are lower bounds on the recall.

All of the systems perform very well on the start time and end time slots, although RAPIER with semantic classes performs significantly worse on start time than the other systems. These two slots are very predictable, both in contents and in context, so the high performance is not surprising. Start time is always present, while end time is not, and this difference in distribution is the reason for the difference in performance by Naive Bayes on the two slots. The difference also seems to impact SRV's performance, but RAPIER performs comparably on the two, resulting in the words only version having apparently better performance on the end time slot than the two CMU systems. WHISK's regular expressions appear not to handle these fields as well as the other systems do.

Location is a somewhat more difficult field and one for which POS tags seem to help quite a bit. This is not surprising, since locations typically consist of a sequence of cardinal numbers and proper nouns, and the POS tags can recognize both of those consistently. This is the one field where WHISK outperforms RAPIER. It is difficult to compare SRV's performance to WHISK and RAPIER. SRV has higher recall but lower precision, and without precise recall numbers, which system is better cannot be easily determined. However, it is clear that all of the relational systems are better than Naive Bayes on this slot, despite the fact that building names recur often in the data and thus the words are very informative.

The most difficult slot in this extraction task is the speaker. This is a slot on which Naive Bayes, WHISK, and RAPIER with words only perform quite poorly, because speaker names seldom recur through the dataset and all of these systems are using word occurrence information and have no reference to the kind of orthographic features which SRV uses or to POS tags, which can provide the information that the speaker names are proper nouns. RAPIER with POS tags performs quite well on this task, with worse recall than SRV, but considerably better precision. Since we don't know the precise recall of SRV, it is difficult to determine which of these systems is better on this slot.

System	acquired			purchaser			seller		
	Prec	Rec	Cov	Prec	Rec	Cov	Prec	Rec	Cov
RAPIER	57.3	19.2	33.2	50.0	19.2	36.4	32.4	10.0	17.7
RAP-WT	58.0	22.1	37.6	51.1	20.5	37.6	33.2	11.0	18.1
RAP-W	54.5	16.8	30.6	51.9	19.6	35.2	41.0	10.0	14.1
NAIBAY	19.8	$\geq$ 19.8	100	$\geq$ 36.9	36.9	100	$\geq$ 15.6	15.6	100
SRV	38.4	$\geq$ 37.1	96.6	42.4	$\geq$ 40.8	96.3	16.4	$\geq$ 13.6	82.7
	acqabr			purchabr			sellerabr		
	Prec	Rec	Cov	Prec	Rec	Cov	Prec	Rec	Cov
RAPIER	43.6	18.5	35.0	42.8	16.7	32.9	10.5	7.3	13.5
RAP-WT	49.5	21.0	36.2	43.8	20.0	38.0	16.4	7.0	15.2
RAP-W	45.5	14.8	27.9	36.43	11.9	26.4	24.83	3.97	7.19
NAIBAY	23.2	$\geq$ 23.2	100	39.6	$\geq$ 39.6	100	16.0	$\geq$ 16.0	100
SRV	31.8	$\geq$ 31.7	99.8	41.4	$\geq$ 41.2	99.6	14.3	$\geq$ 13.6	95.1
	acqloc			status			dlramt		
	Prec	Rec	Cov	Prec	Rec	Cov	Prec	Rec	Cov
RAPIER	46.9	16.3	25.6	67.3	29.8	37.4	63.3	28.5	36.4
RAP-WT	46.5	18.5	27.8	64.8	28.9	37.3	67.4	26.0	31.6
RAP-W	44.3	10.5	16.3	68.8	26.0	32.4	70.0	24.5	28.7
NAIBAY	7.0	$\geq$ 7.0	100	33.3	$\geq$ 33.3	100	24.1	$\geq$ 24.1	100
SRV	12.7	$\geq$ 10.6	83.7	39.1	$\geq$ 35.1	89.8	50.5	$\geq$ 46.0	91.0

Table 4.3: Results for corporate acquisitions task

In general, in this domain semantic classes had very little impact on RAPIER's performance. Semantic constraints are used in the rules, but apparently without any positive or negative effect on the utility of the rules, except on the start time slot, where the use of semantic classes may have discouraged the system from learning the precise contextual rules that are most appropriate for that slot.

All in all, RAPIER performs quite well on this dataset, achieving an F-measure of 82.60% on the overall task using words and POS tags only. Clearly, these results support the usefulness of relational learning for this task as well, although they show that it is not always necessary, since for some slots, such as the times, a propositional system using simple features may be sufficient.

4.8 Corporate Acquisitions Results

As with the seminar announcements domain, for the corporate acquisitions domain, we ran experiments with the versions of RAPIER and compared to figures reported for Naive Bayes and SRV (Freitag, 1998c). Again, the figures are reported by individual slot. Training and testing were performed on 10 random splits of the data into 300 training and 300 testing examples.

Table 4.3 show the results for the five systems on nine slots from this task. Freitag does not report results for the other four slots identified in the data. Again, the figures for recall for Naive Bayes and SRV are lower bounds.

As is immediately clear from the results, this task is much more difficult than the seminar announcements for all of the systems. The six slots which extract names of companies, distinguishing between full and abbreviated names, are extremely difficult for RAPIER. Part of the problem is the distinction between full and abbreviated company names, because it makes it difficult for

Rapier to learn the contexts in which the various fields should be extracted, since the full name of the purchaser and the abbreviated name of the purchaser may show up in very similar contexts, but must be distinguished from one another. The sparseness of data for the seller and seller abbreviation clearly affects all of the learning systems, since all do much more poorly on those fields than on the fields for the purchaser or the entity being acquired. However, the performance of Rapier improves on the two seller slots when it has access to POS tags, showing again that the tags can help generalization when the system does not have much data available to it.

Determining which of the learning systems is best for this task is difficult. Clearly, POS tags help Rapier's performance, not surprisingly as the documents are more grammatical than those in the other domains and they do not rely on visual layout to convey information. It is also clear that Naive Bayes performs worse than SRV on most fields, though not on the seller fields. Comparing SRV and Rapier is problematic with this data. Clearly, in some cases, such as the purchaser abbreviation, SRV is better. In others, such as the acquisition location, Rapier performs better. Because of the lack of exact recall figures and the opposing bias regarding precision (Rapier has a strong bias toward high precision; SRV uses a threshold and can thus have a precision-recall tradeoff, but the figures shown here are intended to have maximal coverage and thus have low precision and higher recall), it is very difficult to say which system performs better on many of the slots shown. However, we can clearly see, again, that relational learning is more effective over all than the Naive Bayes system.

It is clear, from these results, that none of the learners handle this task sufficiently well for useful purposes. It may be that, in order to perform well on this task, Rapier will require some way of representing syntactic phrases and their relationships.

4.9 Discussion

The results above show that relational learning can learn useful rules for information extraction, and that it is more effective than the propositional system used for comparison. Differences between the various relational systems are probably due to two factors. First, the three systems have quite different algorithms, whose biases may be more or less appropriate for particular information extraction tasks. Second, the three systems use different representations and features. All of the systems use word occurrence, and all are capable of representing the sequence of terms. However, Rapier and SRV are capable of representing information about the lengths of fillers (or, in Rapier's case, in other segment of text represented in the rule) without constraining anything else about the filler, and Whisk cannot. Two versions of Rapier make use of POS tags, which the other systems could use, but did not in these experiments. SRV uses Boolean features which provide orthographic information, and neither of the other systems has this information (though in some cases the POS tags provide similar information: capitalized words are usually tagged as proper nouns; numbers are tagged as cardinal numbers). Many of the features used seem quite specific to the seminar announcements domain. It would be useful in the future to examine the effect of the various features, seeing how much of the difference between the relational learning depends upon the different features they use and how much is due to their various algorithmic biases. All of the systems are fairly easily extensible to include the features used by the other systems, so comparing the systems' performance when using the same feature sets should be possible.

Besides looking at the accuracy of a learning system (or in the area of information extraction, its precision and recall), it is important to consider other performance measures, such as training time, testing time, and rulebase complexity.

Because of its expressive, relational representation, RAPIER's training time is fairly long. The following training times are from running RAPIER on an SGI-Origin-200 running IRIX 6.4. For the jobs domain, the words only version varies from an average of 14 minutes to train on 10 examples to an average of 26 hours to train on 270 examples. Increasing the complexity of the learning problem by adding tags increases the training times to 48 minutes to train on 10 examples and 42 hours to train on 270. Training time with semantics are almost identical to those with words and tags only. For the seminar announcements, training time is shorter per example because the examples are shorter and there are fewer slots to learn rules for. Thus, the words only version can train on 240 examples in just under 8 hours on average. Adding tags increases this to an average of 15 hours. The acquisitions data also has fewer slots to train on than the jobs. On 300 examples, the average training time was just under 15 hours for the words only version and 27 hours for the tags version. These training times may seem prohibitive until the time involved for human being to construct similar rules in considered. Development time for information extraction systems is typically measured in man-months. The performance of RAPIER as compared to the Naive Bayes system which trains faster also shows that, on the jobs domain at least, the trade-off between accuracy and speed is in RAPIER's favor.

Lengthy training times are generally not a problem in developing a practical system, but testing times are significant. The testing times in the computer-related jobs domain are under 2 seconds per example without semantics and 3.3 seconds per example using WordNet. The difference is due to the file I/O required to retrieve the WordNet semantic classes. The testing times are lower for the seminar announcements domain and the corporate acquisitions domain which have fewer slots to extract, under 1 second per example for the corporate acquisitions domain with or without semantics and around 8 examples per second without semantics and close to 2 examples per second with semantics in the seminar announcements domain. These times are sufficiently short for the rules to be useful in a practical information extraction system.

One other issue of some interest in a learning system is the complexity of the definition that is learned, and how much that complexity varies with the number of training examples. As an estimate of the complexity of the rule, we use here the same notion of the size of a rule described in Chapter 3. In measuring the size of a rule base, we include only the rules created by the generalization process, not the initial most-specific rules, since the initial rules do not contribute to the rule base's coverage of new examples.

The complexity of the rules seem to depend primarily on the task, being slightly more complex with very small example sets, but not depending much on the type of constraints available. The average rule complexity on the jobs domain is 20 with 10 examples and 19 with more than 100 examples. The complexity of the seminar announcement rules is somewhat higher, averaging 22, and the complexity of the rules for corporate acquisition is higher still, averaging 27. The complexity of the rulebases in the jobs domain grows with the number of examples used to learn the rulebase as does the number of rules. This growth is a little less than straight linear growth, as seen in Figure 4.4.

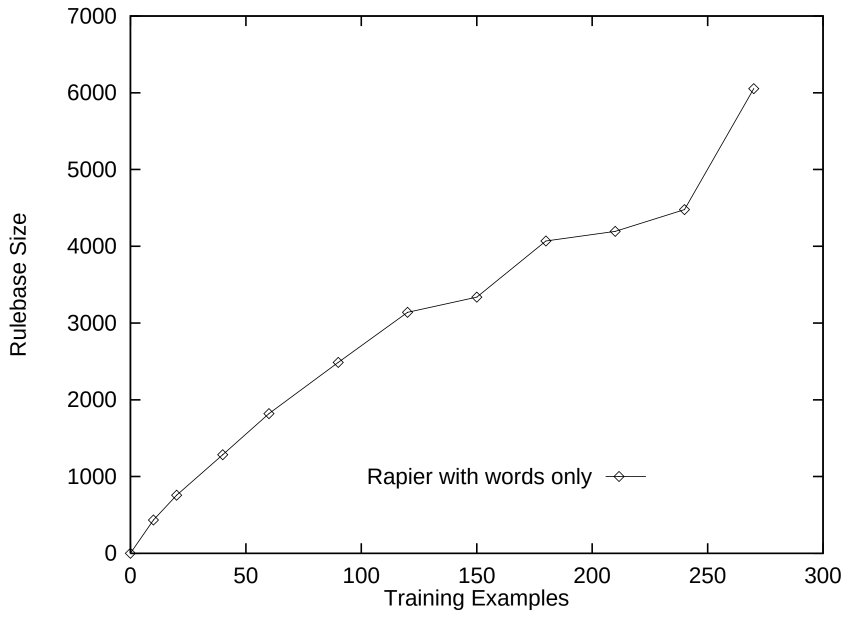

Figure 4.4: Rulebase complexity as a function of the number of examples used to construct the rulebase

4.10 Sample Learned Rules

One final interesting thing to consider about Rapier is the types of rules it creates. One common type of rule learned for certain kinds of slots is the rule that simply memorizes a set of possible slot-fillers. For example, Rapier learns that "mac," "mvs," "aix," and "vms" are platforms in the computer-related jobs domain, since each word only appears in documents where it is to be extracted as a platform slot-filler. One interesting rule along these lines is one which extracts "C++" or "Visual C++" into the language slot. The pre-filler and post-filler patterns are empty, and the filler pattern consists of a pattern list of length 1 with the word constraint "visual" and then pattern items for "c", "+" and "+". In the seminar announcements domain, one rule for the location slot extracts "doherty," "wean" or "weh" (all name of buildings at CMU) followed by a cardinal number. More often, rules which memorize slot-fillers also include some context to ensure that the filler should extracted in this particular case. For example, a rule for the area slot in the jobs domain extracts "gui" or "rpc" if followed by "software."

Other rules rely more on context than on filler patterns. Some of these are for very formal patterns, such as that for the message id of a job posting:

Pre-filler Pattern:	Filler Pattern:	Post-filler Pattern:
1) word: message	1) list: length 5	1) word: >
2) word: -		
3) word: id		
4) word: :		
5) word: <		

Probably the majority of rules have some mix of context and the contents of the filler. An example is the following rule for a title in the computer-related jobs domain:

Pre-filler Pattern:	Filler Pattern:	Post-filler Pattern:
1) word: {:, seeking}	1)	
	2) {consultant, dba}	

In the seminar announcements domain, the following rule for the start time relies on a combinations of the structure of the filler and its context:

Pre-filler Pattern:	Filler Pattern:	Post-filler Pattern:
1) word: {:, for}	1) syntactic: cd	
2)	2) word: :	
	syntactic: :	
	3) syntactic: cd	
	4) syntactic: nn	

In the corporate acquisitions domain, the following rule extracts the status of the acquisition:

Pre-filler Pattern:	Filler Pattern:	Post-filler Pattern:
1) word: {it, executed}	1) word: {agreed, letter}	1) word: {further, to}
2)	2)	
	3)	

Some of the rules are fairly complicated, as is the following rule for the purchaser slot in the corporate acquisitions domain:

Pre-filler Pattern:	Filler Pattern:	Post-filler Pattern:
1) syntactic: {cd, nn}	1) list: length 3	1) word: {said, “,”, “{“}
2) syntactic: {:, to}	syntactic: {:, nnp}	2) syntactic: {prp, dt, nnp}
	2) syntactic: nnp	3)
		4) word: {acquired, investor, for}

For more examples of the types of rules which Rapier learns, see Appendix B.

Chapter 5

Using Active Learning with Rapier

One of the goals of machine learning is to reduce the human time it takes to build a system by having a human being annotate training examples rather than trying to build rules, since the annotation is generally a much easier task, and requires knowledge only of the task to be performed, not of the rule representation. However, even though annotation is easier than building rules in most cases, it can still be quite time-consuming. Therefore, we would like to ensure that our annotation time is spent as effectively as possible.

Examining the learning curves in Chapter 4 for the computer-related job postings domain reveals that the curves are very steep at first and then level off significantly, though they are still climbing slowly. At least part of the reason for this phenomenon is that many of the documents in the training set are quite similar to one another. Once a few examples of a certain class of documents have been seen, seeing more examples of the same class of documents has little effect on the rule base, and the time spent by a person annotating such examples is wasted.

Therefore, we would like to find ways of choosing examples to be annotated that are useful for learning. *Active learning* is the subarea of machine learning concerned with this type of issue. Active learning systems in some way influence the examples they are given, either by constructing examples, by requesting certain types of examples, or by determining which of a set of unlabeled examples are likely to be the most useful. The last option, which is called *selective sampling* (Cohn, Atlas, & Ladner, 1994; Lewis & Catlett, 1994; Dagan & Engelson, 1995), is the most appropriate for a natural language task such as information extraction, since there is a wealth of unannotated data and the only issue is the cost of annotating the data.

5.1 Selective Sampling

The basic idea of selective sampling is that learning begins with a small pool of annotated examples and a large pool of unannotated examples. The learner runs on the annotated examples, and the resulting definition is used to select additional examples to annotate. The selection may proceed in either of two ways: *sequential* sampling or *batch* sampling.

In sequential sampling, the pool of unannotated examples is examined one at a time. The example is labeled according to the current learned definition and then either accepted for learning (in which case, the annotator is asked to label it) or rejected. If the example is selected for learning, the learner updates its definition, and then the next example will be selected.

In batch sampling, the entire pool of examples is labeled using the current definition. The m most useful examples are selected to be annotated and then added to the pool of annotated examples. The learner then updates its definition and again considers the pool of remaining unannotated examples.

There are also two primary methods of determining which examples are most useful: *committee-based sampling* and *uncertainty-based sampling*. 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 unlabeled examples with each definition, and selects for labeling 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 labeling significantly fewer examples compared to random selection. The examples chosen should be those that best guide the learner toward the correct definition, because whenever the definitions disagree about an example, at least one of the definitions must be wrong, and adding that example will help the learning system to produce the correct definition at that particular point. A system may use as few as two committee members, or may use a larger committee, selecting each example for annotation with a probability proportional to the amount of disagreement over its label (Engelson & Dagan, 1996). Committee-based sampling has been successfully used for some natural language tasks such as part-of-speech tagging (Engelson & Dagan, 1996) and text categorization (Liere & Tadepalli, 1997).

In *uncertainty-based sampling*, only one definition is learned, but the definition in some way assigns a confidence measure to each example as it is labeled (Lewis & Catlett, 1994). Then the least confident examples are selected for annotation. The idea here is that the examples which the definition is most confident about will provide very little additional information, while the examples which are least confident will either confirm that the definition is correct and strengthen the confidence or will demonstrate that the definition is not correct and enable the learning to produce a better definition.

5.2 Incorporating Selective Sampling into Rapier

In implementing selective sampling for Rapier, two primary design decision had to be made: choosing between sequential and batch samplings and choosing between committee-based and uncertainty-based sampling.

We chose to use batch sampling because we believe that it is the choice which will more effectively eliminate unnecessary annotation costs. Examining all of the candidates and selecting the best example(s) to annotate is likely to cause performance to increase faster than examining each candidate in turn and accepting or rejecting it. Using batch sampling also avoids the need for establishing thresholds to determine whether the system is sufficiently confident that an example is handled correctly. The system selects for use in learning the example(s) is is least confident about (for uncertainty-based sampling) or about which there is the most disagreement (for committee-based sampling) with no need for concern about how much confidence or how much disagreement there actually is.

For the decision between uncertainty-based and committee-based sampling, there is no a priori reason for believing that either will work better than the other, and neither is likely to be particularly more difficult to implement than the other in the context of Rapier. Uncertainty-based sampling was chosen primarily for efficiency reasons, since using committee-based sampling requires learning multiple rulebases and Rapier's training times are significant, as indicated in chapter 4.

5.2.1 Uncertainty in Rapier

Using uncertainty-based sampling requires a notion of the confidence the system has in its labeling of an example. In order to determine the confidence of the labeling of an example, we need to begin with a notion of the confidence of a rule. Rapier does not have a built-in notion of the certainty of a rule, but, since no thresholding is required, the confidence of a rule can be treated very simply as its coverage of positive examples in the training set with a penalty for coverage of negative examples:

$$conf = pos - 5(neg)$$

. The reasoning behind this notion of confidence is fairly straightforward. It has been shown that the majority of errors made by learned rules are made by rules which cover very small numbers of positive examples (Holte, Acker, & Porter, 1989). Therefore, we want to trust rules that cover large number of positive training examples and distrust those that cover only a few examples. However, we also want to distrust rules which cover negative examples unless the number of positive examples is overwhelming. Thus, we impose a substantial penalty on rules which cover negative examples.

Given this notion of the confidence of a rule, we can determine the confidence of the labeling of a slot. In the case where a single rule finds a slot-filler, the confidence for the slot will be the confidence of the rule that filled that slot. However, when more than one slot-filler is found, the confidence of the slot must be determined. There are three reasonable ways this might be done: 1) take the average of the confidences for the slot-fillers, 2) take the maximum confidence, or 3) take the minimum confidence. Because we want to focuses attention on the least confident rules and find the examples that either confirm or contradict those rules, we choose to use the third alternative.

A final consideration in determining the confidence of each slot is what the confidence of an empty slot is. Since no rule has extracted a slot-filler for the slot, the slot could be considered to have a confidence of 0. However, in some tasks, some slots are empty a large percentage of the time. In the computer-related jobs domain, the salary is present less than half the time, and it is very seldom than both the company and the recruiter appear in the same posting. Thus, we don't want to have an empty salary, company, or recruiter slot necessarily decreasing confidence in the labeling of the example. On the other hand, some slots are always (or almost always) filled, and the absence of slot-fillers for those slots should decrease confidence in an example's labeling. To handle this issue, we keep track of the number of times a slot appears in the training data with no fillers and use that figure as the confidence of the slot when no filler for it is found.

Once the confidence of the slots has been determined, the confidence of an example is easily found by summing the confidence of all slots.

5.2.2 Making Rapier Incremental

One of the major drawbacks to using active learning with Rapier is training time. We would like to select very small pools of examples at each iteration (because selecting several examples at a time may lead us to select several very similar examples), but at each iteration we have to retrain with the additional examples, and training time may begin to become prohibitive. However, making Rapier incremental is a fairly straightforward process which can alleviate this problem.

Rapier's compression approach means changing it to be incremental is fairly simple. Since Rapier begins with a rulebase and then compresses it, instead of creating the rulebase from the examples, it can read in a previously rulebase, add rules for new examples, and then compress the resulting rulebase.

The incremental version of Rapier begins by reading in the old examples (those used to create the old rulebase) and the new examples for which rules are to be added. It then reads in the old rulebase. To prevent the system from keeping rules which are already overly general, Rapier evaluates each rule from the old rulebase on the entire example set. If a rule covers too many negatives (using the same metric and threshold as used for allowing negative coverage described in Section 3.2.3), the rule is removed from the rulebase, and new most-specific rules are created for each positive example covered by the original rule. Finally, the system creates new most-specific rules for each new example. Once the initial rulebase has been developed in this way, learning proceeds exactly as described in Chapter 3.

This incremental version of Rapier does not reduce learning time to a few minutes per example, since new rules must be evaluated on both the old and the new examples (and rule evaluation takes a significant percentage of the learning time) and since it must deal with overly general rules. However, training times are significantly reduced compared to learning from scratch with the old and new examples, making using selective sampling a realistic possibility, and overall performance is not significantly impacted.

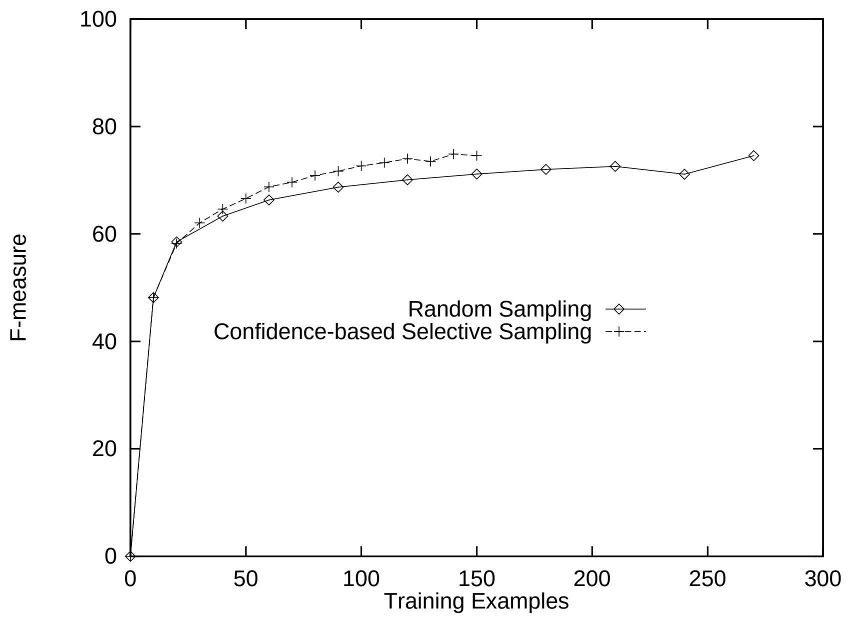

Figure 5.1: Results of selective sampling versus random selection of examples

5.3 Experimental Evaluation

In order to evaluate the utility of using active learning with Rapier we conducted an experiment using the computer-related job posting domain. For this experiment we ran 10 trials, using the same training and testing sets as in Chapter 4. Since Rapier achieved its best results for 270 examples on the jobs domain using word constraints only and using only word constraints is most efficient, we conducted the active learning experiment using word constraints only.

Each trial began with the rulebase constructed from 10 examples in the experiment describe in Chapter 4. Then the selective sampling was carried out with a batch size of one. Thus, the rulebase was run on the remaining 260 examples from the training set, and the example for which the rulebase was least confident was selected for training. Finally, the incremental version of Rapier was used to construct a new rulebase from the original rulebase and the 11 examples, and the process iterated. Precision and recall were measured at 10 example intervals.

Figure 5.1 shows results of the selective sampling process compared to the learning curve using a random example selection from Chapter 4. In order to simplify the comparison between the curves, the F-measure is used. From 30 examples on, the selective sampling consistently outperforms the random example selection. The difference between the curves is not large, but does represent a large reduction in the number of examples required to achieve a given level of performance. At 150 examples, the average F-score is 74.56, exactly the same as the average F-score of the words only version with 270 random examples. This represents a savings of 120 examples, well over one-third of the examples required without selective sampling.

The curve for selective sampling does not go all the way to 270 examples, because once the performance achieved by the random examples at 270 examples is reached, the information available in the data set has been exploited, and the curve will just level off as the less useful examples are added.

These results indicate that a selective sampling method employing a simple confidence measure can, in fact, significantly reduce the number of examples that must be annotated to achieve a given level of performance in this domain.

Chapter 6

Related Work

Although the information extraction task has been worked on for some time, researchers have only recently begun to apply machine learning to the task. A summary of work in the field appeared recently (Cardie, 1997). The earlier attempts to use machine learning worked in the context of propositional machine learning systems, assuming the existence of prior thorough syntactic analysis, and typically depending on a later discourse processing mechanism to complete the task. More recently, with the advent of the world wide web with its wealth of information in textual, but often semi-structured and ungrammatical, form, other researchers have begun to develop systems with the same goals as RAPIER. These systems focus on learning extraction rules whose results can be directly used for a desired task, without dependence on prior parsing or other aspects of a larger information extraction system.

6.1 Early Machine Learning Systems for Information Extraction

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 than 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 list 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 (Huffman, 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 relationships between sentence elements. It finds the relationships that link elements to be extracted, adding non-extracted elements as needed 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.

More recently, a system called ROBOTAG has been developed which uses decision trees to learn the location of slot-fillers in a document (Bennett, Aone, & Lovell, 1997). This system, while newer than others discussed in this section, is similar to these systems in that it is automating part of the information extraction task in the context of a complete information extraction system and it uses propositional learning, specifically, learning decision trees. ROBOTAG uses C4.5 (Quinlan, 1993) to learn decision trees which predict whether a given token is the first token of a slot-filler or the final token of a slot-filler. The features available to the decision trees are the result of pre-processing the text: segmenting the text, performing morphological analysis and lexical lookup. The trees consider the token for which the decision is being made along with a fixed number of tokens around it. Once ROBOTAG has learned trees to identify start and end tokens for slot-fillers, it uses the trees to identify possible start and end tokens and then uses a matching algorithm to pair up start and end tokens to identify actual slot-fillers.

Two primary things distinguish Rapier from these early 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 except RoboTag 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. 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 and RoboTag use 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 as discussed in Section 7.5.

6.2 Learning Information Extraction for Structured Text

One result of the popularity of the World Wide Web has been a need for systems that automatically process web pages. For web pages that contain structured information, it is useful to learn "wrappers" that translate the contents of a web page or class of web pages into a form that would enable software robots to use the pages as if their contents were relational database entries.

One of the first systems designed for this task is Wrapper Induction (Kushmerick, Weld, & Roorenbos, 1997). This system only handles very regular, tabular types of data, since it looks for uniform delimiters that identify the beginning and end of each slot and for delimiters that separate the tabular information from surrounding text. The system only learns rules that cover all of the examples, so any irregularities, such as missing data or fields that are not always in the same order, can keep it from learning a wrapper.

Other systems have since been designed to do wrapper induction, seeking to improve on Kushmerick's system. SoftMealy (Hsu & Dung, 1998) induces finite-state transducers in a bottom-up fashion. Their representation and learning algorithm allow them to handle missing values, slots with multiple values, and variant orderings as will as exceptions and typos–all problems for the earlier system.

Stalker (Muslea et al., 1998) uses a top-down covering algorithm to learn finite automata that act as wrappers. This system is also shown to handle web pages that Kushmerick's system cannot, and to learn from very few training examples.

These systems are all successful and useful systems for their purpose of extracting highly structured data. However, they lack flexibility, being limited to working with highly structured data.

6.3 Recent Learning for Information Extraction from Semi-structured and Free Text

Recently, a few researchers have begun work on learning systems with goals very similar to RAPIER's. These systems are designed to work on multiple types of text, and they avoid reliance on full syntactic analysis and domain-specific semantics, though some can make use of syntactic and semantic information when it is available.

Most of these systems were presented in Section 4.4. The Naive Bayes based classifier is propositional, but has the same goals as far as voiding reliance on syntactic analysis and domain-specific semantics as well being intended to stand alone rather than requiring post-processing (Freitag, 1997). Freitag has extended this work by combining the Naive Bayes classifier with grammatical induction (Freitag, 1997). This approach uses a variety of features, such as whether a token is capitalized or numeric, in addition to actual tokens. In fact, it uses the same feature set as that employed by SRV. Using grammatical induction combined with the Naive Bayes system enables the system to consider the structure of slot-fillers. However, the system does not use context.

SRV (Freitag, 1998c) and WHISK (Soderland, 1998) were both described in full in Section 4.4. They are, clearly, the most similar systems to RAPIER, being relational learning systems with many of the same design goals. Although the three have quite different rule representations, all of them are capable of representing complex rules, including sequential relationships of tokens, and all of them can handle some forms of syntactic or semantic information. The three systems differ significantly in their learning algorithms. SRV and WHISK are both covering algorithms with primarily top-down search; while RAPIER uses a compression algorithm that is primarily bottom-up with a top-down component. Based on the results presented in Chapter 4, SRV and RAPIER seem to perform about the same, with each having some strengths over the others. Whisk performed a little worse than the other systems on the tasks which it was tested on. As was discussed in Chapter 4, it is not clear how much the differences between the systems are attributable to their algorithms and how much they come from the different types of information used by each system. What is clear is that relational learning is a promising direction for information extraction and further research into the strengths and weaknesses of all three of these systems will aid in the development of more effective information extraction systems.

Chapter 7

Future Work

There are a number of directions in which this work could be extended. First, additional testing is desirable, and several additional domains exist on which the system could be evaluated, including extraction from university web pages (Freitag, 1998a), extraction from rental ads (Soderland, 1998), and additional news story domains such as company management changes (DARPA, 1995). Second, a number of enhancements could be made to Rapier's algorithm and representation, including extending it to deal with additional constraint types and learning patterns that relate slots fillers to one another. The work on active learning could be extended to examine the option of using committees rather than the confidence measure and also to explore the possibility of combining a confidence measure and committees. Next, RAPIER makes assumptions that leave open a few issues in information extraction that should be addressed. It does not deal with the distinction between relevant and irrelevant documents for a task, and it assumes only one case frame per document, clearly a problematic assumption for some tasks. Finally, it would be interesting to explore the applicability of RAPIER's representation and general learning algorithm to other types of natural language problems.

7.1 Additional Test Domains

We have tested RAPIER on three different realistic information extraction tasks. We have shown that it performs quite well on two of the tasks, and that its overall performance seems to be comparable to other current learning systems. However, there are several other data sets available that would test RAPIER's performance on other types of documents or tasks. The sets of university web pages for courses and for research projects compiled at CMU (Freitag, 1998a) would test RAPIER's performance on HTML documents, an obviously interesting source for useful information extraction tasks. Another interesting dataset is the apartment rental ads mentioned above (Soderland, 1998). Successful performance on this particular task would, however, require the extensions to learn multiple slot rules. Finally, testing on the MUC-6 company management changes task (DARPA, 1995) would provide a second "free text" domain and would afford the opportunity to compare RAPIER's performance to that of state-of-the-art hand-built information extraction systems.

7.2 Enhancements to Rapier's Representation and Algorithm

7.2.1 Additional Constraint Types

One obvious extension of Rapier would be to add additional constrain types to the representation. One possibility that we have considered is incorporating morphological information. To do this, we could use the morphological functions in WordNet or some alternate source and then allow for constraints on the roots and inflections of words as well as on the surface form of the word. This would add some complexity to the learning algorithm, as these constraints would give rise to the same generalization issues as the word and tag constraints: needing to generalize using disjunction or simply removing constraints.

Another possibly useful addition to Rapier's representation would be to add constraints which are Boolean features of tokens. SRV uses a number of simple Boolean features which are easy to compute by looking at a token: mostly orthographic features such as whether a word is capitalized, whether it is a single character, whether it is "long" according to a pre-specified limit, whether it is entirely upper case, and so on (Freitag, 1998a). Rapier could be extended to incorporate such features, and as long as they were simple Boolean features, they need add no significant computation, since this is a bottom-up algorithm, so there's not a branching factor to be concerned about and each feature would be constrained to be true, constrained to false, or unconstrained, so there are no issues of disjunction or other multiple possible generalizations to consider.

7.2.2 Negated Constraints

Another possible addition for the algorithm would be to add negated 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 to avoid elements of the spurious fillers. Constraints that 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 potential rules.

Intuitively, negated constraints could be very useful. Soderland also suggests the possibility of adding negated constraints to Whisk's regular expressions (Soderland, 1998).

7.2.3 Syntax

Although Rapier performs quite well on the more informal domains on which it has been tested, computer-related job postings and seminar announcements, it does not do as well on the corporate acquisitions domain, which consists of more standard grammatical text. It seems likely that using information from syntactic analysis would be helpful in this type of domain.

In order to incorporate information from parsing into Rapier, it would be necessary to create an alternative type of pattern element, one which represents a syntactic phrase or clause. This would allow Rapier to deal with a syntactic unit as an entity, but would complicate some aspects of the algorithm. Each of the syntactic pattern elements would need to include all of the pattern elements that make them up. Then, a most specific rule would consist of the highest level of syntactic units provided by whatever sentence analysis was used, and each of those would contain references to the remainder of the hierarchy. Then the system would generalize by syntactic units where possible. The specialization algorithm would need to accommodate the alternative pattern element types, considering the option of adding units at each available syntactic level. Thus, if the immediate left of the current rule in the specialization phase was "The man took the dog for a walk" in the first original rule and was "The girl left the cat in the house," the specialization algorithm would need to consider taking the entire sentences, looking at only the verb phrases, looking at only the prepositional phrases, looking at the final noun phrases "a walk" and "the house," or looking at just the final nouns. Thus, the branching factor of the search would increase.

Full parses would not necessarily be required to provide useful syntactic information to Rapier. One simpler option would be to use a noun phrase extractor, a system that identifies the noun phrases in a text, without attempting a full parse.

There is one aspect of syntax which could not be easily incorporated into Rapier. The rule representation would require significant changes in order to express relationships between syntactic units. A unit could be labeled as a noun phrase functioning as a subject, but the representation would not allow for indicating which verb it is the subject of.

7.2.4 Domain-Specific Semantics

In this research, we attempted to use learning with only the information which could be easily obtained with freely available natural language processing tools. Thus, the only semantic information was from a general lexicon. However, the experiments reported in Chapter 4 indicate that incorporating general semantic classes from WordNet does not improve performance, at least in the tasks considered. Therefore, we believe that it would be helpful to explore the possibilities of incorporating domain specific semantic classes. These might include lists of companies, computer languages, operating systems, month names, states, and countries.

7.2.5 Named-Entity Extraction

Beginning in MUC-6, people began to evaluate systems for accomplishing parts of an overall information extraction task (DARPA, 1995). One of the tasks considered in that evaluation was named-entity extraction, which is identifying the proper names which designate entities that might be of interest such as corporations and people. It might useful to use a named-entity extractor as part of the pre-processing of the text for Rapier. This would simplify the rule learning for slots whose appropriate filler were named entities, such as the recruiter and company slots in the computer-related jobs domain, the speaker slot in the seminar announcements and the purchaser, acquired, and seller slots in the corporate acquisitions domain.

7.2.6 Multiple Slot Rules

Another useful extension to Rapier would be the ability to learn rules which extract fillers for multiple slots. There are two advantages to learning such rules. First, if two slots often appear in a particular relationship to one another is a document, then learning a single rule for both slots may help to focus the search for a good rule. This could be helpful for learning start time and end time in the seminar announcements domain, or for learning purchaser and acquired in the corporate acquisitions domain, or for a position and a company in a management changes domain. There are also certain situations where there are multiple fillers for slots, but the fillers are in some way connected. For instance, in a rental ads domain on which WHISK has been tested (Soderland, 1998), it is common to have different sized apartments at different prices listed in the same ad. To simply extract the numbers of bedrooms and the prices without connecting them is not helpful. Learning rules that extract both the number of bedrooms and the related price can help to solve this problem.

Modifying RAPIER to handle multiple slot extraction rules should be fairly straightforward. A rule would simply have additional patterns: one for each slot-filler to be extracted, patterns between the slot-fillers, and the context patterns before the first slot-filler and after the last slot-filler. The primary issue that might be problematic would be the generalization of patterns in between two slot fillers. These would probably contain useful information, but might be long and of different sizes, causing generalizing a pair of them to be prohibitively expensive. It might be necessary to take a more top-down approach to learning the connecting patterns: starting with an unconstrained pattern list and breaking it up into pattern items or smaller lists and adding constraints (taken from those in the original rule pair to limit search) as long as such constraints improved rule quality.

7.2.7 Precision and Recall Trade-offs

In developing an information extraction system, there is often a necessity to trade-off precision and recall. In developing RAPIER we have focused on developing a system with very high precision and reasonably good recall. For the jobs database which partially motivated this work, such a preference seems very appropriate. However, there are tasks where a preference for recall as opposed to precision would be better. If, for instance, the purpose of the extraction was to filter the information for a human who needed all of the information, but didn't want to look at the entire documents, it might be more appropriate to have maximal recall at the cost of lower precision.

RAPIER does not currently have any way to trade off between precision and recall. This could, however, be done in different ways. First, multiple rulebases could be learned using different random seeds. Then recall could be raised by accepting everything extracted by any rulebase, or precision could be raised by accepting only the fillers that were agreed upon by all, or some percentage of the rulebases. This would provide a limited sort of tradeoff.

A second possibility would be to use an approach similar to that taken by SRV, using internal cross-validation or some other method to assign confidences to rules, possibly allowing the creation of more general rules, and then using a threshold to determine how confident the predictions are required to be to vary precision and recall. Even without using cross-validation, this could be done with the notion of the confidence of a rule described in Chapter 5 and used there for selective sampling. This could only be used to further raise the precision of the rulebase, since it does not allow for covering more than the initial rulebase.

Finally, some sort of partial matching could be done with RAPIER's rules, allowing the system to extract items that are "near misses." This could, of course, only be used to raise recall.

7.2.8 Boosting and Bagging

There has recently been quite a bit of work in machine learning on raising the accuracy of classifiers using *boosting* (Freund & Schapire, 1995, 1996) and *bagging* (Breiman, 1998). Both of these methods involve learning multiple definitions of a concept, using example sets that are somehow varied and then having the multiple definitions vote on the classification of new examples.

In boosting, each example has a weight associated with it at each iteration. Initially, a definition is learned with all of the examples equally weighted. Then the weights on the examples are adjusted, increasing the weights of examples that are misclassified by the initial definition and decreasing the weights of correctly classified examples. The boosting process then iterates, learning a new definition with the newly weighted examples and adjusting the weights. The final classifier is the result of using all of the learned definitions to vote, with the weight of each definition's vote being a function of that definition's accuracy on the training set.

In bagging, each definition is learned from a training set of size N which is drawn (with replacement) from the initial set of training examples of the same size. Thus, each training set from which a definition is learned will typically include some examples multiple times and exclude other examples. The final classifier is the result of voting all of these definitions, breaking ties arbitrarily.

Bagging and boosting have both been shown to be effective ways of improving classifier performance (Quinlan, 1996). Thus, it might be useful to try using them to improve learning for information extraction. The one drawback of employing these techniques would be the necessity of learning multiple definitions, since RAPIER's training time is significant.

7.3 Experimenting with Active Learning

The experiment with applying active learning to RAPIER discussed here shows that selective sampling can be usefully applied to learning for information extraction. However, there are still some interesting questions to explore here. As mentioned in Chapter 5, there was no necessity for choosing uncertainty-based selective sampling rather than committee-based. Which would be the more effective choice is an empirical question.

While developing a large committee of rulebases might be too computationally expensive, a committee of two could determine how many of the slots in an example they disagree on and request annotation for the example which produces the most disagreement. As with the confidence-based sampling described in Chapter 5, one problem which needs to be dealt with is unfilled slots. Because of RAPIER's high bias toward precision at the cost of recall, having the two rulebases agree that a slot should be unfilled is not sufficient to indicate that it should, in fact, be empty. Therefore, a committee-based active learning system should have a mechanism to deal with this problem, perhaps randomly counting matching empty slots as a disagreement based on the percentage of times the slot is filled in the training data.

Actually developing differing committee members in RAPIER may be quite easy because of the inherent randomness in the algorithm. If simply using different random seeds proved insufficient, it might be necessary to encourage one rulebase to be more general than the other, either by using different noise parameters or by forcing greater compression of the initial rulebase.

7.4 Open Issues in Information Extraction

7.4.1 Set-valued Slot Fillers

Like other systems for learning extraction patterns, 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 mentioned in Chapter 2 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. However, one could 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.

Another possible way to deal with this issue would be to treat it as a text categorization problem and use a learner designed to handle text categorization per se (Sahami, 1998) rather than adapting RAPIER to the problem. For each set-valued slot, we could use the text categorization learner to learn categorization rules treating each possible value of the slot as a category. However, one potential advantage of using RAPIER and its representation rather than a standard bag-of-words text categorization approach is that, in some cases at least, the information upon which a decision should be based is very localized, and so having the ability to create rules anchored somewhere in the document and looking at a localized context may be more appropriate than a bag of words.

7.4.2 Document Relevance

One issue in information extraction which RAPIER does not deal with is that of identifying relevant and irrelevant documents. In the experiments presented here, all of the documents for training and testing are relevant for the task: i.e. all documents in the computer-related job postings domain are job postings about a computer-related job; all documents in the corporate acquisitions domain do describe an acquisition. However, a fully automated system will have to deal with both relevant and irrelevant documents, or with choosing the relevant template to use: e.g. 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.

Although 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 relevance feedback techniques (Salton, 1989; Frakes & Baeza-Yates, 1992) or to use a text categorization learning system 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.

7.4.3 Recognizing Multiple Case Frames

Along with assuming that each document it sees is relevant, Rapier assumes that each document should produce a single filled template or case frame. This assumption is true of all of the training and test data used in experiments described in this thesis. However, it is not necessarily the case in all relevant documents. For example, in the job postings domain, sometimes a single posting may describe several jobs. In the rental ads domain mentioned above, a single ad may describe include several different apartments at different prices.

The need to create and fill multiple templates for a single document raises several issues. First, the system needs to recognize the need to create multiple templates. One way to do this is to recognize when slots which should have only one filler have multiple fillers extracted. This could be very effective in a domain such as the rental ads. It is less so in a domain like the job postings where almost any slot can have multiple fillers–even job titles may appear in multiple variations for the same job.

Another option would be to recognize the typical ordering of slots, if there are typical orderings. Then, if all fillers for slot A typically come before all fillers for slot B and in a document there are fillers for slot A followed by fillers for slot B followed by more fillers for slot A, this would be a clue that multiple templates should be created.

Yet another option would be to learn text segmentation rules, either using Rapier or applying an alternative learning approach. For a domain like the job postings it may be possible for Rapier to learn rules for recognizing the transition from one job to another since the primary information about the two jobs is seldom intertwined and there are often clear demarcations between jobs. Recently, there has been some research into using learning approaches to segment text (Beeferman, Berger, & Lafferty, 1997; Richmond, Smith, & Amitay, 1997). However, these particular approaches may not be useful to this problem, since they often depend on the shift in vocabulary due to a shift in subject, and within one of these documents, a move from one job to another or from one seminar announcement to another is not likely to produce a change in vocabulary.

A second issue of dealing with multiple case frames is associating the correct fillers with each of the templates. As mentioned above, for some domains this may be facilitated by learning rules which extract fillers for multiple slots. For domains like job postings, it may be possible to simply divide the document into the sections describing each separate case and to apply rules only within each section.

The final issue is probably the simplest for the domains mentioned: recognizing the slots for which only a single value will appear in the document, but that value should apply to all templates created. Examples of this include the message id, posting date, and recruiter or company for a job posting. This issue can probably be handled by simply learning which slots for any given domain will appear only once per document despite applying to all cases.

7.4.4 Template Merging and Coreference Issues

Another issue which arises in some information extraction tasks is that of handling coreference issues. Rapier has avoided these issues, which do not need to be addressed in a task like creating the jobs database, since coreference problems seldom arise, and when they do, simply putting both references in the database is probably appropriate, since the references are likely to be two versions of the job title. However, looking at the seminar announcement task, we can see situations where handling coreference is desirable. For example, the start time of a seminar may be expressed both as “2:00 pm” and as “2pm.” Since these two items are logically the same, our performance measurement counts the slot correctly filled if either appears, and if both versions appear, the slot is still considered to have been correctly filled once, but we don’t penalize the system for extracting both versions. However, in an application, it might be much better to recognize, if both versions are extracted, that the two versions are referring to the same thing and to use only one of them.

7.5 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). It would be interesting 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.

Chapter 8

Conclusions

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. As the amount of textual information available on-line grows, information extraction systems will become more and more important as a method of making this information available and manageable. Manually constructing such information extraction systems is a laborious task; however, learning methods have the potential to help automate the development process. This dissertation has presented a novel rule representation and a relational learning algorithm for learning information extraction rules.

Rapier 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. The system differs from other learning systems attempting the same task in its representation and its primarily bottom-up learning algorithm. Rapier also differs from previous learning systems for information extraction in that it learns rules to accomplish the task without reliance on previous syntactic analysis and that it is intended to be able to handle some information extraction tasks on its own, rather than being a part of a larger information extraction system.

Experimental evaluation of Rapier has shown that it performs well on two realistic information tasks. Its performance is significantly better than a propositional machine learning system for information extraction, supporting our hypothesis that relational learning is more appropriate than propositional learning for this task, due to the structure of natural language. Comparisons with other, very recent relational learning systems for information extraction also support this hypothesis. All three of the relational systems perform well, though further experimentation is required to determine whether the variations in performance are primarily due to the algorithmic differences between the systems or to the different features the systems use.

One of the drawbacks to using supervised machine learning to build information extraction systems is the necessity for a number of annotated examples. Therefore, this research has also focused on a method for reducing the number of examples required to learn useful information extraction rules. Experiments with using selective sampling with Rapier have shown that selective sampling can significantly reduce the amount of annotation required to develop a useful information extraction system with Rapier. Using a simple notion of the confidence of a rule, we achieved with 150 selected examples performance equivalent to a system built with 270 randomly selected examples, reducing the required number of annotated examples by more than a third.

In conclusion, this dissertation has shown that current machine learning techniques, particularly relational learning techniques, in appropriate combinations and used with an appropriate rule representation, can effectively accomplish a useful natural language processing task.

Appendix A

Sample Data

This appendix contains pairs of documents and templates from the domains used in the experiments in this research. The templates consist of the template name followed by a list of slots. Each slot has the slot name followed by a colon and then the slot-fillers, separated by backslashes.

A.1 Computer-related Jobs

Below are two examples of documents and their templates from the computer-related jobs information extraction task.

Document

```
Path: cs.utexas.edu!news-relay.us.dell.com!jump.net!news-fw!news.mpd!
newsgate.tandem.com!su-news-feed1.bbnplanet.com!su-news-hub1.bbnplane
t.com!cpk-news-hub1.bbnplanet.com!news.bbnplanet.com!news-peer.sprint
link.net!news.sprintlink.net!Sprint!ix.netcom.com!news
From: hktexas@ix.netcom.com (Hall Kinion)
Newsgroups: austin.jobs
Subject: US-TX-Austin  WINDOWS, C++, MFC/OWL NEEDED  (tab)
Date: Fri, 29 Aug 1997 19:11:09 GMT
Organization: Netcom
Lines: 22
Message-ID: <34081eca.8972712@NNTP.IX.NETCOM.COM>
NNTP-Posting-Host: aus-tx23-14.ix.netcom.com
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-NETCOM-Date: Fri Aug 29  2:23:27 PM CDT 1997
X-Newsreader: Forte Agent .99f/16.299
Xref: cs.utexas.edu austin.jobs:120178

US-TX-AUSTIN  WINDOWS, C++, MFC/OWL NEEDED (tab)

Excellent opportunity to work with a small team within and
well-established company the develops software for the banking
industry!  New management means new directions and exciting changes.
```

```
A strong Windows C++ Developer is needed to complete a development
team.
Requirements:
Windows 16 and 32 bit
Expert C++
MFC or OWL
Must have done commercial applications
2-5 years experience
Must be a team player, be enthusiastic and ready to learn new
technology.
Salary- $50-70k, DOE

Please respond immediately to:  Tracy
tab@hallkinion.com
ph. 512-349-0960
fax.512-349-0983
```

Template

```
computer_science_job
id: 34081eca.8972712@NNTP.IX.NETCOM.COM
title:
salary: $50-70k
company:
recruiter:
state: TX
city: AUSTIN
country: US
language: C++
platform: WINDOWS
application:
area:   MFC \ OWL
req_years_experience: 2
desired_years_experience: 5
req_degree:
desired_degree:
post_date: 29 Aug 1997
```

Document

```
Path: cs.utexas.edu!cpk-news-hub1.bbnplanet.com!cam-news-hub1.bbnplan
et.com!news.bbnplanet.com!news-feed1.tiac.net!news-master.tiac.net!ne
ws@tiac.net
From: jcookinc@tiac.net (jcookinc)
Newsgroups: austin.jobs,tx.jobs,prg.jobs,comp.jobs,comp.jobs.offered
Subject: SOFTWARE DEVELOPER/PRODUCT DEVELOPER, Austin, TX
Date: Thu, 11 Sep 1997 15:45:29 GMT
Organization: Cook & Associates, Inc.
Lines: 53
```

```
Message-ID: <5v93n8$q2k@news-central.tiac.net>
Reply-To: jcookinc@tiac.net
NNTP-Posting-Host: jcookinc.tiac.net
X-Newsreader: Forte Free Agent 1.0.82
Xref: cs.utexas.edu austin.jobs:122622 tx.jobs:377734

POSTING #445

Major Software Tools vendor located in Austin, Texas has opening
for a Software Developer/Product Developer.

$425 million/annual sales, 1400 employees, private offices, casual
work environment, non-smoking environment (test given during
pre-employment physical), great bonuses, 100match on 401K plan,
leading edge technology.

Will be responsible for Windows development, producing quality code
that is easy to maintain and well documented.  Will perform
functional component design and communicate design considerations to
other team members and product authors. Will perform unit testing
and coordination for problem resolution with QA, support reps, tech
writers, product authors, and managers.

Requires 2+ years Windows development experience, SDK level required,
"C" coding.  Pluses would be any C++, MFC, knowledge of RDBMS
(Oracle, Sybase, Informix, DB2/2, DB2/6K).

Hiring Salary Range $57K to $86K, depending on qualifications.

Excellent benefits and relocation assistance paid by hiring company.

Must have US CITIZENSHIP or GREEN CARD status and actual work
experience.

Strongly prefer reply by E-Mail using MS-WORD, WordPerfect 7 (as
attached file) or ASCII text format. Please refer to Posting Number
in reply.

These are Permanent/Fulltime/Salaried positions NOT CONTRACTs.

Reply To:       John Cook
                COOK & ASSOCIATES, INC.

                Voice: 781-934-6571
                E-Mail: jcookinc@tiac.net


COOK & ASSOCIATES, INC. is a permanent placement firm in business
since 1976. We specialize exclusively in the placement of Information
```

```
Systems/Data Processing professionals on a national and local basis.
A large portion of our business involves relocating systems
professionals within the domestic United States. Our client companies
are diverse and include user organizations of all sizes, proprietary
software vendors, and many Fortune 500 firms.  Let our years of
experience and history of professional, confidential service go to
work for you on your next career move.  Of course, all associated
costs and fees are paid for by our client companies. For more
information on this or other opportunities please E-Mail your resume.
```

Template

```
computer_science_job
id: 5v93n8$q2k@news-central.tiac.net
title: SOFTWARE DEVELOPER/PRODUCT DEVELOPER
salary: $57K to $86K
company:
recruiter: COOK & ASSOCIATES, INC
state: Texas \ TX
city: Austin
country:
language: C++ \ C
platform: Windows
application: DB2/6K \ DB2/2 \ Informix \ Oracle \ Sybase
area:   RDBMS \ MFC
req_years_experience: 2+
desired_years_experience:
req_degree:
desired_degree:
post_date: 11 Sep 1997
```

A.2 Seminar Announcements

Below are two sample documents paired with templates from the seminar announcements domain (Freitag, 1997).

Document

```
Type:     cmu.cs.robotics
Who:      Daniela Rus, Cornell University
Topic:    FINE MOTION PLANNING FOR DEXTEROUS MANIPULATION
Dates:    29-Jan-93
Time:     3:30 PM - 5:00 PM
Place:    ADAMSON WING Auditorium in Baker Hall
PostedBy: maa+ on 26-Jan-93 at 13:23 from ISL1.RI.CMU.EDU (Michelle
Agie)
Abstract:
```

```
          ****************************************************

                                RI SEMINAR

 WHEN:     Friday, Jan 29, 1993; 3:30 pm - 5:00 pm
          Refreshments will be served by 3:15 pm

 WHERE:    ADAMSON WING Auditorium in Baker Hall

 SPEAKER:  Daniela Rus, Cornell University

 TITLE:    FINE MOTION PLANNING FOR DEXTEROUS MANIPULATION

         *Those wishing to meet and talk with Daniela Rus can
          schedule an appointment with Phyliss Pomerantz, by
          calling 7897 or sending e-mail to plp@cs

Dexterity is an important feature for robots that
operate intelligently and independently in their environment.
While planning dexterous manipulation can be viewed as a general
motion planning problem, this leads to intractable algorithms.
Instead, we develop efficient geometric algorithms for the class
of parts orientations problems. For a given set of cooperating agents
(which can be robot fingers, robot arms, mobile robots, or fixtures
in the environment), an object, and a desired reorientation, we wish
to synthesize a  robust plan for the agents that accomplishes the
desired reorientation.  We present an efficient and exact
$O(n\log n)$ algorithm for the reorientation of polygonal objects of
size $n$ and show its extension to polyhedra. This algorithm exploits
the geometric structure of the problem and the task mechanics and
is near-sensorless, in that it requires only sparse sensing.

We are currently implementing the planar reorientation problem in the
context of a team of cooperating autonomous mobile robots.
The team will reorient boxes with polyhedral cross sections.
Since the reorientation plan requires a model, we describe an
algorithm for the robust acquisition of geometric models by mobile
robots with error bounds and demonstrate its performance with a
video.  This work (joint with Jim Jennings) is an exploration
of how, even with sparse and noisy sensors typically found
on mobile robots, we can build adequate accounts and detailed models
of manipulable objects, while satisfying the modeling and information
requests of near-sensorless manipulation planners.
```

Template

seminar
stime: 3:30 PM

```
etime: 5:00 PM
location: ADAMSON WING Auditorium in Baker Hall
speaker: Daniela Rus
```

Document

```
Type:     cmu.cs.scs
Topic:    Special AI Seminar
Dates:    29-Sep-94
Time:     3:00 PM
PostedBy: valdes+ on 28-Sep-94 at 13:22 from CS.CMU.EDU (Raul
Valdes-Perez)
Abstract:
Ken Forbus of Northwestern University will give a seminar on

 "Articulate Virtual Laboratories for Engineering Education"

tomorrow (Thursday) at 3pm in Wean 4625.  The abstract will be
posted when available.
```

Template

```
seminar
stime: 3:00 PM \ 3pm
etime:
location: Wean 4625
speaker: Ken Forbus
```

A.3 Corporate Acquisitions

Below are two sample documents paired with templates from the corporate acquisitions domain (Freitag, 1998c).

Document

```
ENTERTAINMENT {EM} MAY SEEK CRAZY EDDIE {CRZY}
     WASHINGTON, June 29 - Enetertainment Marketing Inc and its
president Elias Zinn have demanded a list of Crazy Eddie Inc
shareholders from the company and said they may pursue a merger
of the Edison, N.J. electronics retailer.
    In a filing with the Securities and Exchange Commission,
Zinn said the demand for the shareholder list was made on June
26 because he may desire to communicate with other Crazy Eddie
shareholders "regarding the affairs" of the company.
    Zinn and his firm, which disclosed they hold a 5.1 pct
stake in Crazy Eddie common stock, said they may acquire more
shares through a negotiated merger or tender offer.
    Entertainment Marketing was informed on June 25 by Shearson
```

Lehman Brothers Inc., acting on behalf of Crazy Eddie, that it would be provided with "certain information" about Crazy Eddie, it told the SEC.

Entertainment Marketing, a Houston-based firm involved in electronics wholesaling and televised home shopping sales, proposed an eight dlr a share merger acquisition of Crazy Eddie on May 29, and modified the proposal on June 9 to include the possible participation of Crazy Eddie management.

Entertainment Marketing told the SEC it expects to meet with Crazy Eddie representatives in the near future.

Entertainment Marketing also disclosed that it retained Drexel Burnham Lambert Inc as its financial advisor and investment banker.

In light of a June 17 announcement from Crazy Eddie that Chemical Bank would no longer fund a 52 mln dlr credit facility with the company, plus further declines in the price of its stock, Entertainment Marketing and Zinn said they are "continuing to evaluate their alternatives with respect to their investment" in Crazy Eddie stock.

Depending on its evaluation of the company, including actions by Crazy Eddie's board and any possible third party bids for the company, Entertainment Marketing and its president said they may hold their present stake in the company, sell some of their shares, or purchase more shares on the open market, through private purchases or in connection with a merger or tender offer.

According to the SEC filing, Entertainment Marketing and Zinn bought their current holdings of 1,560,000 Crazy Eddie common shares between May 20 and June 17 at 7.42 dlrs to 7.93 dlrs a share, or a total of about 11.9 mln dlrs.

Reuter

Template

```
acquisition
purchaser: Enetertainment Marketing Inc
purchabr: ENTERTAINMENT \ Entertainment Marketing
purchcode: EM
acquired: Crazy Eddie Inc
acqabr: CRAZY EDDIE
acqcode: CRZY
seller:
sellerabr:
sellercode:
dlramt:
acqloc: Edison, N.J.
acqbus: electronics
status: may pursue a merger
```

Document

```
AMERICAN TRAVELLERS {ATVC} TO MAKE ACQUISITION
    WARRINGTON, Pa., March 18 - American Travellers Corp said
it has entered into an agreement to purchase ISL Life Insurance
Co of Dallas, a corporate shell with active licenses to operate
in 12 states, for about 400,000 dlrs.
    The company said closing is expected by late spring and
will result in American Travellers being licensed in seven new
states.
 Reuter
```

Template

```
acquisition
purchaser: American Travellers Corp
purchabr: AMERICAN TRAVELLERS
purchcode: ATVC
acquired: ISL Life Insurance Co
acqabr:
acqcode:
seller:
sellerabr:
sellercode:
dlramt: 400,000 dlrs
acqloc: Dallas
acqbus:
status: entered into an agreement
```

Appendix B

Learned Information Extraction Rules

This appendix contains a set of learned information extraction rules for each of the three domains reported on in this dissertation. The rulebases include only the rules created in the compression loop, not any initial, most-specific rules not covered by later, more general rules. These rules were learned in the experiments described in Chapter 4, using Rapier with words and part-of-speech tags.

The format in which the rules are presented here is Prolog-like, in the form:

rule(*TemplateName*, *SlotName*, *NumPosCovered*, *NumNegCovered*,
    Pre-fillerPattern, *FillerPattern*, *Post-fillerPattern*).

Each of the patterns is formatted as a Prolog list of pattern elements. The elements are in the form:

item(*WordConstraints*, *TagConstraints*)
list(*Length*, *WordConstraints*, *TagConstraints*)

where the constraints are "_" if empty, a single constant if there's one constraint, and a Prolog list if the constraint has more than one disjunct.

B.1 Computer-related Jobs

The following rulebase was learned from 270 examples in the experiments described in Section 4.6.

```
rule(computer_science_job, title, 4, 0,
     [item(['junior',':'],_)],[item(['sqa','programmer'],'nnp')],
     [item(_,_),item(_,['nn','sym'])]).
rule(computer_science_job, title, 4, 0,
     [item(_,['dt','sym'])],[item(['programmer','developer'],'nn')],
     [item(['with','date'],_)]).
rule(computer_science_job, title, 6, 0,
     [item(_,['endsent',':']),item(_,_),item([':','sas'],_)],
     [list(1,'gui','nnp'),item('programmer','nnp')],[item(_,['endsent','nnp'])]).
rule(computer_science_job, title, 5, 0,
     [item(['austin','subject'],_),item([',',':'],_)],
     [item(_,'nnp'),item(['developers','dba'],_)],[]).
rule(computer_science_job, title, 3, 0,
     [item(_,[':','endsent'])],[list(2,_,['nns','nnp','nn']),
     item(['consultant','administrators'],_)],[item('endsent','endsent')]).
rule(computer_science_job, title, 4, 0,
     [item(_,'nnp'),item([',','-'],_),item(_,'nnp')],
     [list(2,['/','technical','developer'],_),item('programmer','nnp')],
     [list(2,_,_),item(['recruiter','$'],_)]).
rule(computer_science_job, title, 3, 0,
     [],[item(['visual','software'],'nnp'),item(_,'nnp'),item(_,['nns','nnps'])],[]).
rule(computer_science_job, title, 3, 0,
     [item(['nt','access'],_)],[item(['developers','programmer'],_)],[]).
rule(computer_science_job, title, 4, 0,
     [item(_,['endsent','cd'])],[list(3,_,['nn','nnp','jj']),item('engineer','nn')],
     [item(_,['nnp','vbn'])]).
rule(computer_science_job, title, 3, 0,
     [],[item(_,'nnp'),item(_,'nnp'),item(['manager','specialists'],_)],[]).
rule(computer_science_job, title, 4, 0,
     [item(_,['vbp','nnp'])],[item('programmers','nns')],[item(['with','needed'],_)]).
rule(computer_science_job, title, 3, 0,
     [item(_,['in','nn'])],[item(['lead','software'],_),item(['developers','engineer'],_)],
     [list(2,_,_),item(_,['nn','prp$'])]).
rule(computer_science_job, title, 3, 0,
     [item(['subject','121818'],_),item(_,[':','endsent'])],
     [item(['dba','programmer'],['nnp','jj'])],[]).
rule(computer_science_job, title, 7, 0,
     [item(_,['in',':','vbn'])],[item(['programmers','dba','developer'],_)],
     [item(_,['in','pos','wp'])]).
rule(computer_science_job, title, 4, 0,
     [item(':',':')],[item(_,_),item(['developer','engineer'],_)],[item(_,['cd','prp'])]).
rule(computer_science_job, title, 5, 0,
     [item(_,['jj',':'])],[item(['compiler','peoplesoft','database'],_),item(_,_)],
     [item(_,['in',':','cd'])]).
rule(computer_science_job, title, 3, 0,
     [item(['oracle','cobol'],'nnp')],[item(['dba','programmers'],'nnp')],
     [item('to','to')]).
rule(computer_science_job, title, 4, 0,
     [],[item(_,'nnp'),item(['&','driver'],_),item(['analysts','developer'],_)],[]).
rule(computer_science_job, title, 4, 0,
     [],[item(['network','senior'],['nnp','jj']),item(['security','software'],_),
     item(_,['nnp','nn'])],[]).
rule(computer_science_job, title, 3, 0,
     [item(_,[':','endsent']),item(_,_),list(2,_,_),item(_,['nn',':'])],
     [item(['multimedia','dba'],'nnp'),list(2,['programmer','developers','/'],_)],
     [item(_,['endsent',':']),item(_,['nn','nnp']),item(_,_),item(_,['nn','nnp'])]).
rule(computer_science_job, title, 5, 0,
     [item(_,[':','dt'])],[list(2,_,'nnp'),item(['database','interface'],'nnp'),
     item(['developer','designers'],_)],[]).
rule(computer_science_job, title, 8, 0,
     [item(['java',':'],_)],[list(2,_,['nn','nnp']),item('engineer','nn')],
     [item(_,_),item(_,['in','nn']),item(_,['nnp',':'])]).
rule(computer_science_job, title, 3, 0,
     [item(['a','-'],_)],[item(_,['nn','nnp']),item(['manager','programmers'],_)],
     [item(_,['wp',':'])]).
rule(computer_science_job, title, 4, 0,
    [item(_,['endsent',':'])],[list(3,_,['nn','nnp']),
     item(['engineers','tester'],['nns','nnp'])],[item(_,['.','endsent'])]).
rule(computer_science_job, title, 6, 0,
     [item(['-','+'],_)],[list(2,['/','mainframe','programmer'],_),
     item(['programmer','analyst'],_)],[item(['needed','endsent'],_)]).
rule(computer_science_job, title, 4, 0,
     [item(_,['endsent',':'])],[item(_,'nnp'),item(['qa','driver'],'nnp'),
     item(_,'nn')],[]).
rule(computer_science_job, title, 5, 0,
     [],[item(['device','senior'],_),list(2,_,_),item(['developer','programmer'],_)],
     [item('endsent','endsent')]).
rule(computer_science_job, title, 3, 0,
     [],[item(_,'nnp'),item(['intranet','sqa'],'nnp'),item(_,_)],
     [item(_,['endsent','vbn'])]).
rule(computer_science_job, title, 3, 0,
     [item(_,[':','endsent'])],[item(_,'nnp'),item(['programmer','manager'],'nnp')],
     [item('-',':')]).
rule(computer_science_job, title, 5, 0,
     [],[item(['software','network','senior'],_),list(2,_,'nnp'),
     item(['administrator','admin'],_)],[]).
rule(computer_science_job, title, 5, 0,
     [],[item('programmer','nnp'),item(['analysts','analyst'],_)],
     [item(_,['nn','endsent'])]).
rule(computer_science_job, title, 4, 0,
     [list(2,_,_),list(2,_,_),item(_,['endsent','sym'])],
     [list(2,_,['nnp','nn']),item(['developers','engineer'],_)],
     [item(_,['endsent','('])]).
rule(computer_science_job, title, 7, 0,
     [item(_,'nn'),item(_,_),item(_,_),item(['endsent',':'],_)],
     [list(2,_,'nnp'),item(['developers','engineer'],_)],[]).
rule(computer_science_job, title, 4, 0,
     [item(_,[':','endsent'])],[list(2,_,['nns','nnps','jj']),
     item('programmer','nnp')],[item('endsent','endsent'),
     item(['req','description'],_)]).
rule(computer_science_job, title, 5, 0,
     [item([':','/'],_)],[list(1,'senior','nnp'),item('software','nnp'),
     item(['consultant','engineer'],_)],[]).
rule(computer_science_job, title, 3, 0,
[],[item(['senior','human'],_),item(_,'nnp'),item(_,'nnp'),item(_,'nn')],[]).
rule(computer_science_job, title, 3, 0,
     [item(_,':')],[item(['software','infrastructure'],'nnp'),
     list(2,_,'nnp'),item(['manager','analysts'],_)],[]).
rule(computer_science_job, title, 3, 0,
     [item(_,':')],[list(2,_,'nnp'),item(['analyst','technician'],_)],
     [item('endsent','endsent')]).
rule(computer_science_job, title, 5, 0,
     [item(_,':')],[item(_,'nnp'),item(['administrator','engineer'],_)],
     [list(2,['location','for','endsent'],_),item(_,['nnp',':'])]).
rule(computer_science_job, title, 4, 0,
     [item(['austin','assertive'],_),item(['/','customer'],_)],
     [list(2,_,'nnp'),item(['engineers','developer'],_)],
     [item(_,['endsent','to'])]).
rule(computer_science_job, title, 3, 0,
     [item(['/','-'],_)],[item(_,'nnp'),item(['engineer','master'],_)],
     [item(_,_),item(_,_),item(_,[':','nnp']),item(_,['nnp','nns'])]).
rule(computer_science_job, title, 3, 0,
     [item(_,':')],[item(_,['jj','nnp']),item(['c','support'],'nnp'),
     item(_,'nnp')],[]).
rule(computer_science_job, title, 3, 0,
     [item(['-','basic'],_)],[list(3,['sr.','(','mainframe'],_),
     item('programmer','nnp'),list(1,'analysts','nns')],
     [item(_,[':','endsent'])]).
rule(computer_science_job, title, 3, 0,
     [item(['2200','several'],_)],
     [list(2,['analysts','programmers','systems'],_)],
     [item(_,['.','vbp'])]).
rule(computer_science_job, title, 3, 0,
     [item(['for','austin'],_)],[list(2,_,'nnp'),
     item(['engineers','executives'],'nns')],[]).
rule(computer_science_job, state, 140, 0,
     [list(3,_,_)],[item(['ohio','tx','california'],'nnp')],[]).
```

```
rule(computer_science_job, state, 80, 1,
     [],[item(['tx','oh'],_)],[list(2,_,_),item(_,_),item(_,'nnp'),
     item(_,['to','nnp'])]).
rule(computer_science_job, state, 138, 0,
     [list(3,_,_)],[item(['mi','ohio','tx'],'nnp')],
     [item(_,[',',':','endsent'])]).
rule(computer_science_job, state, 136, 0,
     [list(3,_,_)],[item(['oh','tx'],_)],[]).
rule(computer_science_job, state, 161, 2,
     [list(2,[',','endsent','dallas'],_)],[item(['tx','az'],'nnp')],[]).
rule(computer_science_job, state, 143, 0,
     [item(_,[':','in']),list(2,_,_)],[item(['tx','ohio'],'nnp')],[]).
rule(computer_science_job, state, 136, 0,
     [list(2,_,_),item(_,[':',','])],[item(['oh','tx'],_)],
     [item(['-','78746'],_),item(_,['nnp','cc'])]).
rule(computer_science_job, state, 132, 0,
     [list(2,_,_)],[item(['tx','co'],'nnp')],[item(['75248','-'],_)]).
rule(computer_science_job, state, 109, 0,
     [item(_,['jj',':']),list(2,_,_),item(_,[':',','])],
     [item(['ohio','tx'],'nnp')],[item(_,[':','endsent']),
     list(2,[',','duration','progress'],_),item(_,[':','nnp'])]).
rule(computer_science_job, salary, 4, 0,
     [item(_,['nn','sym']),list(2,_,_),item(_,[':','nnp'])],
     [list(2,_,['cd','to','$']),item(_,_),item('65k','cd')],[]).
rule(computer_science_job, salary, 3, 0,
     [item(_,[':','endsent']),list(2,_,_)],[item(['to','40'],_),item(_,_),
     item(['55','50'],_),list(2,_,_),item(_,['cd','.'])],[]).
rule(computer_science_job, salary, 4, 0,
     [],[list(1,'to','to'),item('$','$'),list(2,_,_),item(_,_),item(_,_),
     item(['120k','yr'],_)],[]).
rule(computer_science_job, salary, 7, 0,
     [],[item('$','$'),item(_,'cd'),list(2,['35','to','-'],_),
     item(_,['$','nn']),item(['78k','hr'],['cd','nn'])],[]).
rule(computer_science_job, salary, 4, 0,
     [item(_,[':','in','nns'])],[list(1,'$','$'),item('to','to'),
     item('$','$'),item(['65k','90k','75k'],'cd')],[]).
rule(computer_science_job, salary, 11, 0,
     [],[list(1,'to','to'),item('$','$'),
     item(['45k','62k','85k','60k'],'cd')],[]).
rule(computer_science_job, salary, 5, 0,
     [item(['60',':','administrator'],_)],[item(_,['to','$']),
     list(3,_,_),item(['65','60k','hr'],_)],[]).
rule(computer_science_job, salary, 3, 0,
     [item(_,['in','vb'])],[list(2,_,['cd','$']),
     item(['figures','pa'],_)],[]).
rule(computer_science_job, salary, 4, 0,
     [item(['us','intranet','endsent'],_)],[list(3,_,['cd','to','$']),
     item('$','$'),item(['20','40'],'cd'),list(2,_,_),
     item(['hr','hour'],_)],[]).
rule(computer_science_job, salary, 4, 0,
     [],[item('to','to'),item('$','$'),item(['55k','95k','40k'],_)],[]).
rule(computer_science_job, salary, 11, 0,
     [item(['up','programmer'],_)],[item('to','to'),item('$','$'),
     list(3,_,[',','cd'])],[item(_,['in','endsent'])]).
rule(computer_science_job, salary, 13, 0,
     [],[item('$','$'),list(3,_,[',','cd']),item(_,_),item('$','$'),
     item(_,'cd'),item(_,_),item(_,['nnp','cd'])],[]).
rule(computer_science_job, salary, 4, 0,
     [item(_,[':','nnp'])],[list(1,'to','to'),item('$','$'),
     item(['43k','40'],'cd'),item(_,_),item(_,['$','nnp']),item(_,_)],[]).
rule(computer_science_job, salary, 7, 0,
     [item(_,['in','cc'])],[list(1,'to','to'),item('$','$'),
     list(3,_,[',','cd']),item('/','nn'),item(_,'nn')],[]).
rule(computer_science_job, salary, 6, 0,
     [],[item('$','$'),list(2,_,_),item(_,_),item(['90k','70k'],'cd')],[]).
rule(computer_science_job, salary, 3, 0,
     [item(_,['in',':'])],[item(_,['to','$']),list(2,_,_),
     item(['85k','72k'],'cd')],[]).
rule(computer_science_job, req_years_experience, 4, 0,
     [list(2,['-','for','environment'],_)],[item(['5','4'],_),
     item('+','sym')],[]).
rule(computer_science_job, req_years_experience, 13, 0,
     [item(['have','with'],_)],[item(_,'cd'),item('+','sym')],[]).
rule(computer_science_job, req_years_experience, 9, 0,
     [item(['from',',','*'],_)],[item(['1','5','2'],_)],
     [item(_,['nns',':'])]).
rule(computer_science_job, req_years_experience, 3, 0,
     [list(2,_,_),item(['endsent','opportunities'],_),list(2,_,_)],
     [item(['two','5'],_)],[item(_,['cc',':'])]).
rule(computer_science_job, req_years_experience, 4, 0,
     [item(_,['cd','in']),item(_,['(','jjs'])],[item('2','cd')],
     [item(_,['sym','nns']),item(_,['nns','endsent'])]).
rule(computer_science_job, req_years_experience, 7, 0,
     [],[item('3','cd'),item('+','sym')],[item('years','nns'),
     item(_,['vbp','vbg'])]).
rule(computer_science_job, req_years_experience, 6, 0,
     [item(_,['cc','endsent'])],[item(['2','five'],'cd')],
     [item('years','nns')]).
rule(computer_science_job, req_years_experience, 3, 0,
     [item(['required','position'],_),item(_,_)],[item(['3','2'],'cd')],
     [item('-',':')]).
rule(computer_science_job, req_years_experience, 9, 0,
     [item(_,['endsent','vb'])],[item(['3','5'],_),item('+','sym')],[]).
rule(computer_science_job, req_years_experience, 8, 0,
     [item(['endsent','of'],_)],[item(['two','2'],'cd')],
     [item(_,['cc','nns'])]).
rule(computer_science_job, req_years_experience, 3, 0,
     [item(_,[',','nnps'])],[item(['2','4'],'cd')],
     [item(_,['nns','endsent'])]).
rule(computer_science_job, req_years_experience, 9, 0,
     [item(['endsent','mpeg'],_)],[item(['3','2'],'cd')],
     [item(['-','software'],_)]).
rule(computer_science_job, req_years_experience, 9, 0,
     [item(_,['jjs','vb'])],[item('3','cd'),list(1,'+','sym')],
     [item(_,'nns')]).
rule(computer_science_job, req_years_experience, 12, 0,
     [list(2,['endsent','-',':'],_)],[item('3','cd'),list(1,'+','sym')],
     [item(['to','years'],_)]).
rule(computer_science_job, req_years_experience, 8, 0,
     [item(_,['vb','endsent'])],[item('2','cd'),list(1,'+','sym')],
     [item(_,'nns'),list(2,_,_),item(_,['in','nnp'])]).
rule(computer_science_job, recruiter, 5, 0,
     [],[item(['computemp','lcs','unasyst'],_)],
     [item(_,['in',':','endsent'])]).
rule(computer_science_job, recruiter, 16, 0,
     [item(_,'nnp'),item('endsent','endsent')],
     [item(['information','cadre','mccoy'],_),item(_,_),item(_,_)],[]).
rule(computer_science_job, recruiter, 3, 0,
     [item(_,['.',':']),item('endsent','endsent')],
     [item(['omega','online'],'nnp'),item(_,'nnp')],[]).
rule(computer_science_job, recruiter, 7, 0,
     [list(2,_,_),item([':','endsent'],_)],
     [list(3,['&','employment','global','search','strategic'],_),
     item(['staffing','international'],'nnp')],[]).
```

```
rule(computer_science_job, recruiter, 3, 0,
     [item(_,['cd','nn']),list(2,_,_)],[item(['lcs','esi'],'nnp')],
     [item(_,[',','pos'])]).
rule(computer_science_job, recruiter, 53, 1,
     [item(['taranto','0120'],_),item(_,_)],[item(_,'nnp'),
     item(['placement','spectrum'],'nnp')],[]).
rule(computer_science_job, recruiter, 46, 1,
     [item(_,['prp','.']),item(_,_)],[item(['technology','resource'],_),
     item(['resources','spectrum'],'nnp')],[item(_,[':','endsent'])]).
rule(computer_science_job, post_date, 260, 2,
    [],[item(_,'cd'),item(_,'nnp'),item(['1997','97'],_)],[]).
rule(computer_science_job, post_date, 168, 0,
     [],[item(_,_),item('sep','nnp'),item('1997',_)],[]).
rule(computer_science_job, platform, 29, 0,
     [list(2,_,_)],[item(['unix','rhetorex'],'nnp')],[]).
rule(computer_science_job, platform, 6, 0,
     [],[item('os','nnp'),item('/','nn'),item('2','cd')],[]).
rule(computer_science_job, platform, 28, 1,
     [item(_,[':','cc','('])],[item(['as400','nt','ibm','mvs'],'nnp')],[]).
rule(computer_science_job, platform, 9, 0,
     [],[item(['hp3000','mvs'],'nnp')],[]).
rule(computer_science_job, platform, 4, 0,
     [],[item(['windows95','os400'],_)],[]).
rule(computer_science_job, platform, 3, 0,
     [],[item(['8086','vax'],_)],[]).
rule(computer_science_job, platform, 3, 0,
     [],[item(['rs6000','sparc'],'nnp')],[]).
rule(computer_science_job, platform, 15, 0,
     [],[item(['aix','windowsnt'],'nnp')],[]).
rule(computer_science_job, platform, 4, 0,
     [],[item(['as','multi'],_),item(_,_),item(['400','workstation'],_)],
     []).
rule(computer_science_job, platform, 5, 0,
     [],[item(['6502','novell'],_)],[item(_,[',','endsent'])]).
rule(computer_science_job, platform, 5, 0,
     [item(['(','in'],_)],[item(['windows','ms'],'nnp'),
     item(_,['cd','nnp'])],[]).
rule(computer_science_job, platform, 28, 1,
     [],[item(_,'nnp'),item(['ux','95'],_)],[]).
rule(computer_science_job, platform, 15, 0,
     [list(2,['systems','will','operating'],_),item(_,['vb',':'])],
     [item(['nt','solaris'],'nnp')],[]).
rule(computer_science_job, platform, 8, 0,
     [item(_,['nn','in'])],[item(['windows','pc'],_)],[item(_,'nns')]).
rule(computer_science_job, platform, 35, 0,
     [],[list(1,_,'nnp'),item('nt','nnp')],[item(_,['endsent',','])]).
rule(computer_science_job, platform, 30, 1,
     [list(2,_,_)],[item(['mac','solaris','unix'],'nnp')],[]).
rule(computer_science_job, platform, 18, 0,
     [item(_,['in','nn'])],[item(['dos','nt'],'nnp')],[]).
rule(computer_science_job, platform, 4, 0,
     [item(['as','('],_)],[item(['dialogic','windows'],'nnp')],[]).
rule(computer_science_job, platform, 7, 0,
     [],[item(['win32','sun'],'nnp')],[item(['api',','],_)]).
rule(computer_science_job, platform, 48, 2,
     [item(_,['nn',',','nnp','cc'])],[item(['win95','unix','amd29k'],_)],
     []).
rule(computer_science_job, platform, 8, 0,
     [item(['unix','in'],_),item(_,[',','dt'])],[list(2,['/','ms','os'],_),
     item(['windows','2'],_)],[item(_,[',','nn']),item(_,['cc','.'])]).
rule(computer_science_job, platform, 13, 0,
     [],[item(['unix','sun'],'nnp')],[item(_,_),item(_,['vbz','nnp']),
     item(_,_),item(_,_),item(_,['.',','])]).
rule(computer_science_job, platform, 6, 0,
     [item(_,[':','cd'])],[item(['aix','nt'],'nnp')],[]).
rule(computer_science_job, platform, 43, 0,
     [],[item('windows','nnp'),item(['nt','3.1'],_)],[]).
rule(computer_science_job, platform, 3, 0,
     [item(_,[',','sym'])],[item(['winnt','x86'],_)],[]).
rule(computer_science_job, platform, 4, 0,
     [],[item(['mac','hp'],'nnp')],[item(['web','and'],_)]).
rule(computer_science_job, platform, 6, 0,
     [item(_,['nn','dt']),list(2,['landmark','+'],_)],
     [item(['aix','nt'],'nnp')],[item(_,['nn','vbd'])]).
rule(computer_science_job, platform, 13, 0,
     [item(_,['nn','dt'])],[item(['win','unix'],'nnp')],[]).
rule(computer_science_job, platform, 3, 0,
     [item(['strong','y2k'],_)],[item(['dos','ibm'],'nnp')],[]).
rule(computer_science_job, platform, 3, 0,
     [item(_,['vb','nn'])],[item(['bsd','95'],_)],[]).
rule(computer_science_job, platform, 5, 0,
     [item(_,['nn','endsent'])],[item(['solaris','windows'],'nnp')],
     [item(_,['nn','nnp'])]).
rule(computer_science_job, platform, 18, 0,
     [item(_,[',','nn','in'])],[item(['nt','aix','as400'],'nnp')],
     [item(_,_),item(_,['nnp','cd'])]).
rule(computer_science_job, platform, 4, 0,
     [],[item(_,'nnp'),item(['2200','pc'],_)],[]).
rule(computer_science_job, platform, 34, 0,
     [item(_,['nnp','cc']),item(_,_)],[item(['unix','x86'],_)],[]).
rule(computer_science_job, platform, 16, 0,
    [item(_,['nns','nnp']),item(',',',')],[item(['unix','psos'],_)],[]).
rule(computer_science_job, platform, 34, 1,
     [item(_,['in',','])],[list(2,_,[':','nnp']),
     item(['nt','ux'],'nnp')],[]).
rule(computer_science_job, language, 14, 0,
     [list(2,['knowldge','mfc','working'],_),item(_,_)],[item('visual','jj'),
     item('basic','nnp')],[]).
rule(computer_science_job, language, 19, 0,
     [list(2,[',','intel','powerbuilder'],_)],[item(['x86','visual'],_),
     item(['assembly','basic'],_)],[]).
rule(computer_science_job, language, 26, 0,
     [item(['-',','],_)],[list(2,['/','pro','microfocus','*','cobol'],_),
     item(['c','cobol','400'],_)],[item(_,['to',','])]).
rule(computer_science_job, language, 8, 0,
     [],[item(['vc','rpg'],_),item(_,_),item(_,['sym','cd'])],[]).
rule(computer_science_job, language, 44, 0,
     [item(_,['in',',','nn','('])],[item(['rpgiii','pb','natural','fortran',
     'c','ksh'],_)],[item(_,['nn','cc','nns',','])]).
rule(computer_science_job, language, 19, 0,
     [list(2,['using','or','development'],_)],[item(['oracle','visual'],_),
     item(['forms','basic'],_)],[]).
rule(computer_science_job, language, 12, 0,
     [item(_,['endsent','nnp']),item(_,_)],
     [item(['progress','roscoe','cics','rpgiv'],'nnp')],[]).
rule(computer_science_job, language, 27, 0,
     [],[item(['progress','html','java'],_)],[item([',','programming'],_)]).
rule(computer_science_job, language, 24, 0,
     [item(['rdbms',','],_)],[item(['c','java'],'nn')],[item(_,[',','nn'])]).
rule(computer_science_job, language, 41, 0,
     [],[item(['html','sequel','powerbuilder'],'nnp')],[]).
rule(computer_science_job, language, 14, 0,
     [item(_,['endsent',':','cc'])],[item(['clos','c','tcl'],_)],
     [item(_,[',','nnp','.'])]).
```

```
rule(computer_science_job, language, 57, 1,
     [],[item(['powerbuilder','c','idms'],_)],
     [item(_,['.','nns','cc','nn','to'])]).
rule(computer_science_job, language, 9, 0,
     [item(_,[',','in','vbp']),list(2,_,_),item(_,['cc','.'])],
     [item(['ims','c','cl','pascal'],_)],[list(2,_,_),
     list(2,['db2','modern','.','and','endsent'],_),
     item(_,[',','endsent','dt'])]).
rule(computer_science_job, language, 7, 0,
     [],[item(['bsh','cics'],_)],[list(2,_,_),item(_,['nn','endsent'])]).
rule(computer_science_job, language, 26, 1,
     [],[item(['assembler','delphi','cobol'],_)],[item(_,[',','.'])]).
rule(computer_science_job, language, 53, 0,
     [list(2,['or','oracle','austin','-'],_)],[item(['vc','c','pl'],_),
     item(['+','/'],_),item(['+','sql'],_)],[]).
rule(computer_science_job, language, 36, 0,
     [item(['writer','c',',','understanding'],_),
     item(['/','and','unix','of'],_)],
     [item(['cobol','c','shell','javascript'],_)],[]).
rule(computer_science_job, language, 23, 0,
     [item(_,['nn','nnp','endsent']),list(2,_,_)],
     [item(['idl','c','idms'],_)],[item(_,['nn','to'])]).
rule(computer_science_job, language, 26, 0,
     [item(_,['cc',','])],[list(1,'objective','nnp'),item('c','nn')],
     [item(_,['nn',','])]).
rule(computer_science_job, language, 34, 1,
     [item([',','-','consider'],_)],[item(['rpgii','java','cobol'],'nnp')],
     []).
rule(computer_science_job, language, 21, 0,
     [list(2,[',','-','an','austin'],_)],[item(['java','rpg','pli'],_)],
     []).
rule(computer_science_job, language, 7, 0,
     [],[item(['smalltalk','sql'],'nnp')],[item(_,['nn','.'])]).
rule(computer_science_job, language, 15, 0,
     [],[item(['ims','cobol'],'nnp')],[item(_,[',','endsent']),
     item(_,['nnp','jj'])]).
rule(computer_science_job, language, 38, 0,
     [item(['html','c','internet'],_),item(_,[',','nn'])],
     [item(['xml','c','powerbuilder'],_)],[]).
rule(computer_science_job, language, 18, 0,
     [list(2,[',','of','pascal'],_)],[item(['visual','mq'],_),
     item(['basic','smalltalk'],'nnp')],[]).
rule(computer_science_job, language, 18, 0,
     [],[item('cobol','nnp'),list(1,'ii','nnp')],[item(_,['in',','])]).
rule(computer_science_job, language, 14, 0,
     [list(2,_,_)],[item(['delphi','natural','java','fortran',
     'javascript'],['nnp','nn'])],[]).
rule(computer_science_job, language, 9, 0,
     [],[item('visual','jj'),item('c','nn'),item('+','sym'),
     item('+','sym')],[]).
rule(computer_science_job, language, 24, 0,
     [item(_,['nn',':',','])],[item(['powerbuilder','c','rpg'],_)],
     [item(_,['endsent',',','in'])]).
rule(computer_science_job, language, 17, 0,
     [item(_,[',','vbg','in'])],[item(['pascal','sql','powerbuilder'],'nnp')],
     [item(_,[',','endsent'])]).
rule(computer_science_job, language, 9, 0,
     [item(['ms',','],_)],[list(2,_,['sym','nnp']),item(['shell','+'],_)],
     [item(_,['endsent',','])]).
rule(computer_science_job, language, 42, 0,
     [item(_,[',','in'])],[list(2,['c','vc','visual'],_),
     item('+','sym'),item('+','sym')],[]).
```

```
rule(computer_science_job, language, 3, 0,
     [item(['and','endsent'],_)],[item(['sql','cobol'],_)],[item(_,'nn')]).
rule(computer_science_job, language, 51, 0,
     [item(_,[',',':','in'])],[item(['sql','cobol','c'],_)],
     [item(_,[',','endsent','dt'])]).
rule(computer_science_job, language, 56, 0,
     [item(_,[',','in'])],[item(['assembly','c','sql'],_)],
     [item(_,['nn',',','cc'])]).
rule(computer_science_job, language, 8, 0,
     [item(_,['endsent','in']),list(2,_,_)],
     [item(['rpg400','cics'],'nnp')],[]).
rule(computer_science_job, language, 24, 0,
     [item(_,[':','in'])],[item(['c','pl'],_),item(['+','/'],_),
     item(_,['sym','cd'])],[]).
rule(computer_science_job, language, 14, 0,
     [],[item(['jcl','vb'],'nnp')],[]).
rule(computer_science_job, language, 11, 0,
     [],[item(['java','vbscript'],'nnp')],[item(_,_),item(_,['in','nn'])]).
rule(computer_science_job, language, 5, 0,
     [item(_,[',','cc'])],[list(2,['*','bourne','sql'],_),
     item(['shell','reportwriter'],_)],[]).
rule(computer_science_job, language, 26, 0,
     [],[item(['c','tcl'],_),item(['+','/'],_),item(['+','tk'],_)],[item(_,_),
     item(_,['endsent','jj'])]).
rule(computer_science_job, language, 44, 0,
     [item(_,['in','nn'])],[item(_,_),item('+','sym'),item('+','sym')],[]).
rule(computer_science_job, language, 32, 0,
     [item(_,['endsent',','])],[item(['c','pl'],_),item(['+','/'],_),
     item(_,['sym','cd'])],[]).
rule(computer_science_job, language, 38, 0,
     [item(_,['in',','])],[item(['sql','c'],_),item(['*','+'],'sym'),
     item(_,_)],[]).
rule(computer_science_job, language, 26, 0,
     [item(_,['nn',',']),item(_,['nn','rb'])],[item(['c','delphi'],_)],[]).
rule(computer_science_job, language, 7, 0,
     [],[item(['4gl','shell'],['cd','nn'])],[item(_,['sym','nn'])]).
rule(computer_science_job, language, 32, 0,
     [],[item(['powerbuilder','java'],'nnp')],[item(_,['nn',','])]).
rule(computer_science_job, language, 22, 0,
     [],[item(['cobol','assembly'],'nnp')],[item(['programmers',','],_)]).
rule(computer_science_job, language, 13, 0,
     [],[item(['javascript','perl'],'nnp')],[]).
rule(computer_science_job, id, 263, 2,
     [item('endsent','endsent'),item('message','nn'),item('-',':'),
     item('id','nn'),item(':',':'),item('<','sym')],[list(7,_,_)],
     [item('>','sym')]).
rule(computer_science_job, id, 166, 0,
     [item(':',':'),item('<','sym')],[list(5,_,_)],[item('>','sym'),
item('endsent','endsent')]).
rule(computer_science_job, desired_years_experience, 12, 0,
     [item(['-','to'],_)],[item(['6','5','7'],_)],[item('years','nns'),
     item('experience','vbp'),item(_,[':','in'])]).
rule(computer_science_job, desired_years_experience, 18, 0,
     [item(['-','to'],_)],[item(['8','4','5','10','6'],['nn','cd'])],
     [item('years','nns'),item(_,_),item(_,['nnp','jj','in','nn'])]).
rule(computer_science_job, desired_years_experience, 10, 0,
     [item(['3','4','two'],'cd'),item(_,_)],
     [item(['5','four'],['nn','cd'])],[]).
rule(computer_science_job, desired_years_experience, 20, 1,
     [item('-',':')],[item(['4','5'],_)],[item(_,['nn','nns'])]).
rule(computer_science_job, desired_years_experience, 17, 0,
     [item(_,['vbg','endsent','sym',',']),item(_,'cd'),item(_,[':','to'])],
     [item(['5','4'],_)],[item(_,'nns')]).
rule(computer_science_job, desired_degree, 6, 0,
     [item(_,['endsent',':','cc'])],
     [item(['masters','phd','b.s','mscs'],_)],[]).
rule(computer_science_job, desired_degree, 4, 0,
     [item(['bs','preferred'],_),item(_,['nn','cc',':'])],
     [item(['ba','ms','bs'],'nnp')],[]).
rule(computer_science_job, desired_degree, 5, 0,
     [],[item(['msme','m.b.a'],'nnp')],[]).
rule(computer_science_job, country, 112, 2,
     [item(['in','oracle','and',':','endsent'],_)],
     [item(['uk','usa','us'],['prp','nnp'])],[]).
rule(computer_science_job, country, 109, 2,
     [item(['and',':','endsent'],_)],[item(['uk','usa','us'],['nnp','prp'])],
     []).
rule(computer_science_job, country, 58, 1,
     [item(['78759','pay','join'],_)],[item(['usa','us'],_)],[]).
rule(computer_science_job, company, 5, 0,
     [],[item(['siemens','caleb'],'nnp'),
     list(2,_,'nnp'),item(_,['nns','.'])],[]).
rule(computer_science_job, company, 6, 0,
     [],[item(['cps','s3g','aerotek'],_)],[]).
rule(computer_science_job, company, 32, 0,
     [],[item(['ses','alliance','victina'],'nnp')],[]).
rule(computer_science_job, company, 3, 0,
     [item(_,['.','nn']),item(_,_)],[item(['wesson','edp'],_),
     list(2,_,['nnps','nnp','nn'])],[item(_,['vbz','endsent'])]).
rule(computer_science_job, company, 7, 0,
     [item(_,['nn','nnp','endsent']),item(_,_),item(_,_),
     item(_,['endsent',':'])],[item(['new','momentum','rb'],_),
     list(3,_,_)],[item(_,['vbz','(',','])]).
rule(computer_science_job, company, 3, 0,
     [list(2,['-','cs.utexas.edu','pc'],_),item(_,_),item(_,['nnp',':']),
     item(_,_),item(_,['in','endsent'])],[item(_,'nnp'),
     list(2,['-','consultants','in'],_),item([',','house'],_),
     item(['inc.','design'],'nnp')],[item(_,['.','endsent']),item(_,_),
     item(_,_),list(2,_,_),item(_,['jj','nn']),item(_,_),item(_,['nn','"'])]).
rule(computer_science_job, company, 3, 0,
     [],[item(['austinite','soft'],_),item(_,_)],[]).
rule(computer_science_job, company, 4, 0,
     [item(['with',':'],_)],[list(2,_,'nnp'),
     item(['graphics','terminals'],_)],[]).
rule(computer_science_job, company, 8, 0,
     [],[item(['ctg','ptg'],'nnp')],[]).
rule(computer_science_job, city, 179, 0,
     [list(2,_,_)],[item('austin','nnp')],[]).
rule(computer_science_job, city, 180, 0,
     [list(2,_,_)],[item(['austin','london'],'nnp')],[]).
rule(computer_science_job, city, 64, 0,
     [],[item(['cleveland','austin'],'nnp')],[item(['endsent','tx.'],_)]).
rule(computer_science_job, city, 25, 0,
     [item(_,['rb','in'])],[item('austin','nnp')],[]).
rule(computer_science_job, area, 4, 0,
     [item([',','of'],_)],[list(2,['security','atm','internet'],_)],
     [item([',','with'],_),item(_,['nnp','jj']),item(_,['nnp','nn'])]).
rule(computer_science_job, area, 3, 0,
     [item(['writing','-'],_)],[item(['device','project'],_),
     item(_,['nns','nnp'])],[]).
rule(computer_science_job, area, 3, 0,
     [item(['for','with'],'in')],[item(_,['nn','nnp']),
     item(['2000','mrp'],_)],[]).
rule(computer_science_job, area, 4, 0,
     [item(_,['endsent',':'])],[item(['communications','erp'],_)],[]).
rule(computer_science_job, area, 4, 0,
     [],[item(_,['nnp','nn']),item(['qualification','2000','driver'],_)],
    [item(_,['nnp','nn']),item(_,[':','.','in'])]).
rule(computer_science_job, area, 4, 0,
     [item([':','oracle'],_)],[item(['web','dba'],'nnp')],[item(_,_),
     item(_,['cd','endsent'])]).
rule(computer_science_job, area, 11, 0,
     [],[item(['ole','adsm','rdb'],'nnp')],[]).
rule(computer_science_job, area, 3, 0,
     [item(_,[':','cc'])],[item(['rdbms','database'],_)],
     [item(['/','exposure'],'nn')]).
rule(computer_science_job, area, 3, 0,
     [item(_,[':','endsent'])],[item(['compiler','cgi'],_)],[]).
rule(computer_science_job, area, 3, 0,
     [],[item(['samp','functional'],'jj'),item(_,'nn')],[]).
rule(computer_science_job, area, 3, 0,
     [item(_,['nnp','.']),item(_,_)],[item(['real','tcp'],'jj'),
     item(_,_),item(_,'nnp')],[]).
rule(computer_science_job, area, 3, 0,
     [item(_,['endsent','nn']),item(_,['in','cc'])],
     [item(['gui','network'],_)],[]).
rule(computer_science_job, area, 3, 0,
     [item(['or','for'],_)],[item(['embedded','graphic'],_),
     list(2,['interfaces','systems','user'],_)],[item(['.','experience'],_)]).
rule(computer_science_job, area, 13, 0,
     [],[item(['lan','3d','odbc','ai','realtime'],_)],
     [item(['executables','games','programming','endsent','/'],_)]).
rule(computer_science_job, area, 5, 0,
     [],[item(['wans','opengl'],'nnp')],[]).
rule(computer_science_job, area, 6, 0,
     [list(2,_,_)],[item(['gui','accounting'],_)],[]).
rule(computer_science_job, area, 9, 0,
     [item(_,[',','nn'])],[item(['internet','ood'],_)],
     [item(_,['endsent',','])]).
rule(computer_science_job, area, 13, 0,
     [item(_,['sym',':','nn']),item(_,_)],
     [item(['ooad','mfc','odi'],'nnp')],[]).
rule(computer_science_job, area, 3, 0,
     [item('and','cc')],[item(['modeling','ole'],_)],[]).
rule(computer_science_job, area, 5, 0,
     [],[item(['network','3d'],['nnp','cd']),
     item(['management','graphics'],_)],[]).
rule(computer_science_job, area, 7, 0,
     [],[item(['sqa','wan'],'nnp')],[list(2,_,_),item(_,['cc','nnp'])]).
rule(computer_science_job, area, 3, 0,
     [list(2,_,_),item(_,['in',':'])],[item(['databases','internet'],_)],
     [item(_,_),item(_,['nn','nnp']),item(_,['to','nnp'])]).
rule(computer_science_job, area, 3, 0,
     [item(['direct','and'],_)],[item(_,['nn','nnp'])],
     [item(['and','system'],['cc','nn']),item(_,['nn','nnps']),item(_,_),
     item(_,['.','vbg'])]).
rule(computer_science_job, area, 3, 0,
     [item(_,['cc','nnp'])],[item(['networking','embedded'],['vbg','vbn'])],
     [item(_,[':','nnp'])]).
rule(computer_science_job, area, 4, 0,
     [item(_,['in','cc'])],[item(['web','internet'],_)],[item(_,'nns')]).
rule(computer_science_job, area, 7, 0,
     [list(2,['of','','/','endsent'],_)],[item(['rpc','adsl','cgi'],'nnp')],
     []).
rule(computer_science_job, area, 4, 0,
     [],[item(['networking','automated'],_),
     item(['protocols','testing'],_)],[]).
rule(computer_science_job, area, 5, 0,
     [item(_,[':','vbg']),list(2,_,_),item(_,['vbg',':'])],
     [item(['gui','web'],'nnp')],[]).
rule(computer_science_job, area, 7, 0,
     [list(2,['hot','oracle','endsent'],_)],[item(['dba','games'],'nnp')],
     [item(['endsent','programmer'],_)]).
rule(computer_science_job, area, 6, 0,
     [],[item(['mfc','multimedia'],'nnp')],[item(_,['dt','nns'])]).
rule(computer_science_job, area, 6, 0,
     [],[item(['wan','cgi'],'nnp')],[item(_,['nns','nn'])]).
rule(computer_science_job, area, 6, 0,
     [item(_,'nnp'),item(_,[',','endsent'])],[item(['telecom','odbc'],_)],
     []).
rule(computer_science_job, area, 13, 0,
     [item(_,['nn',','])],[item(['client','network'],_),
     list(2,_,['cc','nn','nnp']),item(['server','technologies'],_)],[]).
rule(computer_science_job, area, 3, 0,
     [item(',',',')],[item(['animation','games'],_)],[]).
rule(computer_science_job, area, 8, 0,
     [item(_,[',',':'])],[item(['mfc','tso'],'nnp')],[]).
rule(computer_science_job, area, 3, 0,
     [item(_,['nn','endsent'])],[item(['dcom','graphics'],'nnp')],[]).
rule(computer_science_job, area, 9, 0,
     [item(_,[',','nn'])],[item(['gui','rdbms'],'nnp')],[]).
rule(computer_science_job, area, 4, 0,
     [],[item(['network','distributed'],_),
     item(['gaming','systems'],['nn','nns'])],[]).
rule(computer_science_job, area, 7, 0,
     [],[item(['corba','com'],'nnp')],[]).
rule(computer_science_job, area, 9, 0,
     [item(_,['nn','nnp']),item(_,['in',','])],[list(2,['/','device','tcp'],_),
     item(['drivers','ip'],_)],[]).
rule(computer_science_job, area, 5, 0,
     [item(_,['nnp',','])],[item(['y2k','dms'],'nnp')],[]).
rule(computer_science_job, area, 4, 0,
     [item(_,':'),item(_,_),item('-',':')],[item(['dba','telecom'],'nnp')],
     [item(_,_),item(_,['cc','endsent'])]).
rule(computer_science_job, area, 12, 0,
     [],[item(['isapi','lan'],'nnp')],[]).
rule(computer_science_job, area, 7, 0,
     [],[item(['failure','device'],'nnp'),item(_,'nnp')],[]).
rule(computer_science_job, application, 5, 0,
     [item(_,[':',','])],[item(['paradox','robohelp','server'],'nnp')],[]).
rule(computer_science_job, application, 28, 1,
     [],[item(['sybase','maestro','domino','easytrieve','adabas'],'nnp')],
     []).
rule(computer_science_job, application, 5, 0,
    [],[item(['dialog','ms','hp'],_),
     item(['manager','frontpage','openview'],_)],[]).
rule(computer_science_job, application, 3, 0,
     [item(_,['nnp','jj']),item(['utilities',','],_)],
     [list(3,['/','access','qa','client'],_),item(['partner','400'],_)],[]).
rule(computer_science_job, application, 53, 2,
     [],[item(['excel','oracle','foxpro','xrunner','debabelizer'],'nnp')],
     []).
rule(computer_science_job, application, 8, 0,
     [item(_,['cc','in'])],[item(['pm','access','clearcase'],_)],[]).
rule(computer_science_job, application, 3, 0,
     [item([',','progress'],_),item(['or',','],_)],
     [list(3,_,['fw','nn','nnp'])],[item(_,['vbz',','])]).
rule(computer_science_job, application, 12, 0,
     [item(_,['nn','endsent']),item(_,_)],[item(['verilog','db2'],_)],[]).
rule(computer_science_job, application, 10, 0,
     [item(_,'nnp'),item(_,[':',','])],
     [item(['informix','netscape'],'nnp')],[]).
rule(computer_science_job, application, 5, 0,
     [],[item(_,'nnp'),item(['back','beans','view'],_)],[]).
rule(computer_science_job, application, 4, 0,
     [item(_,['endsent',','])],[list(2,['-','sql','control'],_),
     item(['server','m'],'nnp')],[item(_,['endsent',',']),item(_,'nnp'),
     item(_,_),item(_,['nnp',','])]).
rule(computer_science_job, application, 4, 0,
     [item(['oracle','access'],_),item(',',',')],[list(2,_,'nnp')],
     [item(_,['cc',',']),item(['sybase','gui'],'nnp')]).
rule(computer_science_job, application, 8, 0,
     [item(_,['endsent','nnp','nns']),item(_,_)],
     [item(['construct','sas','directdraw'],_)],[]).
rule(computer_science_job, application, 24, 0,
     [item(_,['nnp','nn']),item(_,_)],
     [item(['sybase','qad','groupware','backoffice','ideal'],'nnp')],[]).
rule(computer_science_job, application, 33, 0,
     [],[item(['oracle','db2'],_)],[item(_,[',','cd','vbz'])]).
rule(computer_science_job, application, 3, 0,
     [item(_,['nnp','(']),item(['back','java'],'nnp'),item(_,_)],
     [item(['dazzle','cgi'],'nnp')],[item(_,['endsent',','])]).
rule(computer_science_job, application, 17, 0,
     [],[item(['ingres','sybase','endevor'],_)],[list(2,_,_),
     item(_,['nn',',',':'])]).
rule(computer_science_job, application, 4, 0,
     [item(['hacmp','management','cics'],'nnp'),item(',',',')],
     [list(3,['6000','/','hp','db2','openview','netview'],_)],
     [item(_,['endsent','(',':'])]).
rule(computer_science_job, application, 5, 0,
     [item(_,[',','cc'])],[item(_,'nnp'),item(['office','access'],_)],[]).
rule(computer_science_job, application, 10, 0,
     [item(_,['cc','sym','nn'])],[item(_,'nnp'),item('server','nnp')],[]).
rule(computer_science_job, application, 10, 0,
     [],[item(['sap','vsam','ita'],'nnp')],[]).
rule(computer_science_job, application, 6, 0,
     [],[item(['operation','ms'],'nnp'),item(['center','access'],_)],[]).
rule(computer_science_job, application, 4, 0,
     [item([',','with'],_)],[item(['netscape','peoplesoft'],'nnp')],
     [item([',','implementation'],_)]).
rule(computer_science_job, application, 4, 0,
     [item(['heavy','or'],_)],[item(['access','ims'],'nnp')],[]).
rule(computer_science_job, application, 13, 0,
     [],[item(['ims','db2','ges'],_)],[item(_,['cc','nn','dt'])]).
rule(computer_science_job, application, 13, 0,
     [item(_,[',','nnp','in'])],[item(['informix','sap','tso'],'nnp')],[]).
rule(computer_science_job, application, 5, 0,
     [],[item(['essbase','focus','postalsoft'],'nnp')],[]).
rule(computer_science_job, application, 3, 0,
     [item(_,[',',':'])],[item(['cosmic','crystal'],'nnp')],
     [item(_,[',','nnp']),item(_,['cc','nnp'])]).
rule(computer_science_job, application, 20, 1,
     [item(['or',','],_)],[item(['ims','db2'],_)],[]).
rule(computer_science_job, application, 7, 0,
     [],[item(['winrunner','alc'],_)],[]).
rule(computer_science_job, application, 4, 0,
     [],[item(['macromedia','direct'],_),item(['director','3'],_),item(_,_),
     item(_,_)],[]).
rule(computer_science_job, application, 7, 0,
     [],[item(['pop','access','abap'],'nnp')],[item(_,['nns','endsent','nnp'])]).
```

```
rule(computer_science_job, application, 5, 0,
     [],[item(['db','life'],'nnp'),item(_,['nn',':']),
     item(_,['cd','nnp'])],[]).
rule(computer_science_job, application, 7, 0,
     [item(',',',')],[item('access','nn')],[]).
rule(computer_science_job, application, 6, 0,
     [],[item(['s2k','informix'],'nnp')],[item(_,[',','nns'])]).
rule(computer_science_job, application, 4, 0,
     [],[item(['peoplesoft','excell'],'nnp')],[item(['.','access'],_)]).
rule(computer_science_job, application, 6, 0,
     [],[item('lotus',_),item('notes',_)],[]).
rule(computer_science_job, application, 10, 0,
     [item(_,['in','nn'])],[item(['access','sas'],_)],[]).
rule(computer_science_job, application, 27, 0,
     [item(_,[':',','])],[item(['oracle','visualtest'],_)],[]).
```

B.2 Seminar Announcements

The following rulebase was learned in the experiments reported in Section 4.7 using Rapier with words and part-of-speech tags.

```
rule(seminar, stime, 86, 2,
     [list(2,_,_)],[item(_,'cd'),item(':',':'),item(_,'cd')],
     [item(_,[':','.'])]).
rule(seminar, stime, 155, 0,
     [item([':','at'],_)],[item(_,_),item(':',':'),item(['00','30'],'cd'),
     item(['p.m.','am','pm','p.m'],_)],[]).
rule(seminar, stime, 4, 0,
     [item([':','at'],_)],[item(['5pm','12.30'],_)],[]).
rule(seminar, stime, 20, 0,
     [item(_,_)],[item(['3','4','7'],'cd'),item(':',':'),item(_,'cd'),
     item('p.m.',_)],[list(3,_,_),item(_,['nnp',':','dt'])]).
rule(seminar, stime, 27, 0,
     [item(['from','endsent'],_)],[list(2,_,[':','cd','nn']),
     item(['pm','30pm','30'],_)],[item(_,[':','in','.'])]).
rule(seminar, stime, 17, 0,
     [item(['time','.'],_),item([':','endsent'],_)],[list(3,_,[':','nnp','cd'])],
     [item(['endsent','-'],_),item(['place','1'],_)]).
rule(seminar, stime, 6, 0,
     [item('at','in')],[item(_,_),item(':',':'),item('30pm','cd')],[]).
rule(seminar, stime, 26, 0,
     [item(['from',':'],_)],[item(_,'cd'),item(':',':'),
     item(['30pm','30'],'cd')],[item(_,['to','endsent'])]).
rule(seminar, stime, 51, 1,
     [item(_,[',','endsent','cd'])],[item(_,'cd'),item(':',':'),
     list(2,['p.m.','30pm','30'],_)],[item(_,['endsent',',',':'])]).
rule(seminar, stime, 144, 3,
     [item(['at',':',';'],_)],[item(['11','12','2','3'],'cd'),
     item(':',':'),list(2,['pm','00','45','30'],_)],
     [item(_,[',','endsent',':'])]).
rule(seminar, stime, 6, 0,
     [item([':','at'],_)],[item(['9.00','1','5','7'],_),
     item(_,['nn','nnp'])],[item(['in','endsent',','],_)]).
rule(seminar, stime, 3, 0,
     [],[item(['7pm','4pm'],'cd')],[]).
rule(seminar, stime, 4, 0,
     [item(['promptly','1994'],_),item(['at','endsent'],_)],
     [list(3,_,[':','cd'])],[item(['until','-'],_)]).
rule(seminar, stime, 77, 0,
     [item(['time','auditorium'],_),item([':',','],_)],[item(_,'cd'),
     item(':',':'),item(_,'cd')],[item(_,[':','.'])]).
rule(seminar, stime, 3, 0,
     [],[item(['5','4'],_),item('p.m.','nn')],[]).
rule(seminar, stime, 8, 0,
     [],[item(['1','3'],'cd'),item(':',':'),item(['25','30pm'],'cd')],
     [item(_,_),item(_,['nn','endsent'])]).
rule(seminar, stime, 18, 0,
     [item([',',':','seminar'],_)],[item(['12','1'],'cd'),item(':',':'),
     item(_,'cd')],[item(_,[':','in'])]).
rule(seminar, stime, 4, 0,
     [item(_,['in',','])],[list(2,_,_),item(['am','00pm'],['nnp','cd'])],
     []).
rule(seminar, stime, 32, 0,
     [item(_,['nn',',']),item([':','at'],_)],[item(['11','3'],'cd'),
     item(':',':'),item(['30','30pm'],'cd')],[item(_,[':','.'])]).
rule(seminar, stime, 20, 0,
     [item(['at',':'],_)],[item(_,'cd'),item(':',':'),item(_,'cd'),
     item(['a.m.','am'],_)],[]).
rule(seminar, stime, 6, 0,
     [item(['endsent','at'],_)],[list(2,_,_),item(['noon','30pm'],_)],
     [item(_,_),list(2,_,_),item(_,['cc',':'])]).
rule(seminar, stime, 3, 0,
     [item(_,[',','nnp'])],[item('3','cd'),item(':',':'),item(_,'cd')],
     [item(_,['endsent','nnp'])]).
rule(seminar, stime, 23, 0,
     [item(':',':')],[item(['2','10'],'cd'),item(':',':'),item(_,'cd')],
     [item(_,['endsent',':'])]).
rule(seminar, stime, 56, 0,
     [item(_,[',','endsent'])],[item(_,'cd'),item(':',':'),item(_,'cd')],
     [item('-',':')]).
rule(seminar, stime, 10, 0,
     [item(_,['endsent','in'])],[item(_,'cd'),item(':',':'),
     item(['30pm','00'],'cd')],[item(['-','in'],_),item(_,_),
     item([':','7500'],_)]).
rule(seminar, stime, 17, 0,
     [item([',',':'],_)],[item(['2','4'],'cd'),item(':',':'),
     item('00','cd'),list(1,'p.m.','nn')],[item(_,['endsent',':'])]).
rule(seminar, speaker, 5, 0,
     [list(2,_,_)],[item(['terra','toshiaki','mike','steven'],'nnp'),
     item(_,'nnp')],[]).
rule(seminar, speaker, 6, 0,
     [item(_,['in',':']),item(['endsent','roy'],_)],[list(2,_,'nnp')],
     [item(['endsent','will','on'],_),item(['november','be','15'],_)]).
rule(seminar, speaker, 5, 0,
    [item(_,[',','cd']),item(['and','/'],_)],[item(_,'nnp'),item(_,'nnp')],
     [item(_,['(',','])]).
rule(seminar, speaker, 4, 0,
     [],[list(2,_,'nnp'),item(['jones','brezany','mclaughlin'],'nnp')],
     [item(_,_),item(_,['nn','dt','nnp'])]).
rule(seminar, speaker, 4, 0,
     [item(_,['sym','nn','nnp']),item(_,[':','endsent'])],
     [item(['roger','dan','jerome'],'nnp'),item(_,'nnp')],[]).
rule(seminar, speaker, 19, 0,
     [item([':','by','endsent','7220'],_)],[item(['patrick','rajiv','dr.',
     'pat','hugh'],'nnp'),list(3,_,[':','nnp'])],
     [item(_,[',','vbz','endsent'])]).
rule(seminar, speaker, 16, 0,
     [list(2,['.','instructor','"',':','pattern'],_),
     item(_,[':','endsent',','])],
     [item(['keith','leonard','alex','dr.'],'nnp'),list(3,_,'nnp')],
     [item(_,['(','endsent','md',','])]).
rule(seminar, speaker, 6, 0,
     [],[item(['professor','william'],'nnp'),item(_,'nnp'),item(_,'nnp')],
     [item(['of','endsent'],_)]).
rule(seminar, speaker, 13, 0,
     [list(2,_,_),item(_,['endsent',','])],[item(['lev','alex','dr.'],'nnp'),
     list(3,_,'nnp')],[item(_,['endsent','md',','])]).
rule(seminar, speaker, 3, 0,
     [],[item(['richard','eric'],'nnp'),list(3,[',','nyberg','m','h.'],_),
     item(_,_)],[item('endsent','endsent')]).
rule(seminar, speaker, 19, 0,
     [item(_,['in','"','cc',',','endsent',':'])],[list(3,_,'nnp'),
     item(['hager','christianson','trinkle','szeliski','sathi','kedzierski',
     'gunzinger','fedder','baheti','smith','fink','papadimitriou',
     'bracken'],'nnp')],[]).
rule(seminar, speaker, 9, 0,
     [item(_,['endsent','nn'])],[item(['jonathan','chip','dinesh','robert',
     'erik','bruce'],'nnp'),item(_,'nnp')],[]).
rule(seminar, speaker, 3, 0,
     [item(_,_),item(_,_)],[list(2,_,'nnp')],[item(['of','endsent'],_),
     item(['istar','dartmouth'],'nnp')]).
rule(seminar, speaker, 9, 0,
     [item(_,[':','nn'])],[item(['randy','patrick','john','tony','gordon',
     'ralph'],'nnp'),item(_,'nnp')],[]).
rule(seminar, speaker, 3, 0,
     [item(_,[':','nnp']),item('endsent','endsent')],[item(['nobutoshi',
     'professor'],'nnp'),item(_,'nnp')],[item(_,['md','vbz'])]).
rule(seminar, speaker, 6, 0,
     [item(_,['endsent',':'])],[list(3,_,'nnp'),
     item(['gould','lamport','lennert'],'nnp')],[]).
rule(seminar, speaker, 5, 0,
     [],[item(['mel','wojciech','rita'],'nnp'),item(_,'nnp')],[]).
rule(seminar, speaker, 8, 0,
     [item(_,['in','endsent','vbg'])],[item(['alessandro','robert','david',
     'andrew'],'nnp'),item(_,'nnp')],[item(_,_),item(_,['vb','prp','nnp'])]).
rule(seminar, speaker, 8, 0,
     [item(_,['in','cc',':'])],[list(3,_,'nnp'),
     item(['borenstein','smith','simmons','desa'],'nnp')],[]).
rule(seminar, speaker, 12, 0,
     [],[item(_,'nnp'),item(['gregory','koechling','etzioni','skiena',
     'perlant','booch','fukawa','bischof'],_)],[]).
rule(seminar, speaker, 3, 0,
     [item(['by',')'],_),item([':',','],_)],[item(_,'nnp'),item(_,'nnp')],
     [item(_,['endsent','('])]).
rule(seminar, speaker, 3, 0,
     [item(['endsent','and'],_)],[item(_,'nnp'),item(_,'nnp')],
     [item(['endsent','computational'],_),item(['institute','linguistics'],'nnp')]).
rule(seminar, speaker, 4, 0,
     [item('endsent','endsent')],[item(['bennett','paul','kai'],'nnp'),
     list(3,_,['nnp',':'])],[item(_,[',','endsent','cc'])]).
rule(seminar, speaker, 15, 0,
     [],[item(['jim','dr.'],'nnp'),list(3,_,'nnp')],
     [item(_,['in','dt','endsent'])]).
rule(seminar, speaker, 4, 0,
     [item(_,':')],[item(['peter','eric','neil'],'nnp'),item(_,'nnp')],[]).
rule(seminar, speaker, 3, 0,
     [list(2,[':','endsent','who'],_)],[item(['michael','dr.jim'],'nnp'),
     item(_,_)],[item(_,['endsent',',']),item(_,'nnp')]).
rule(seminar, speaker, 3, 0,
     [],[item(['lara','mr.'],'nnp'),list(2,_,'nnp')],
     [item(['biodegradation',','],_)]).
rule(seminar, speaker, 4, 0,
     [item(['mechanisms','5409'],_),item('endsent','endsent')],[item(_,'nnp'),
     item(_,'nnp')],[item('endsent','endsent')]).
```

```
rule(seminar, speaker, 8, 0,
     [item(_,['endsent',',','vb'])],[item(['steve','michelle','ed','fil'],'nnp'),
     item(_,'nnp')],[]).
rule(seminar, speaker, 7, 0,
     [item(_,['endsent',':'])],[item(['tom','james','cindy'],'nnp'),
     item(_,'nnp')],[item(_,['in','endsent','cc'])]).
rule(seminar, speaker, 7, 0,
     [],[item(_,'nnp'),item(['mountford','fitzgerald','amber','munter',
     'hancock','dario'],'nnp')],[]).
rule(seminar, speaker, 22, 0,
     [item(['endsent','biomolecules'],_),item(['speaker','"'],_),
     item([':','endsent'],_)],[item(_,'nnp'),item(_,'nnp')],
     [item(_,['endsent',','])]).
rule(seminar, speaker, 6, 0,
     [],[item(['michael','mr.'],'nnp'),list(2,_,'nnp')],
     [item(_,['endsent','('])]).
rule(seminar, speaker, 6, 0,
     [],[item(['laura','mr.','alex'],'nnp'),item(_,'nnp')],
     [item(_,['vbz','endsent'])]).
rule(seminar, speaker, 4, 0,
     [item(_,['cc','in',':'])],[item(['rick','mr.','victor'],'nnp'),
     item(_,'nnp')],[]).
rule(seminar, speaker, 4, 0,
     [],[item(_,'nnp'),item(['adelson','christel','kelly'],'nnp')],[]).
rule(seminar, speaker, 4, 0,
     [],[item(['ralph','ido','maxine'],'nnp'),list(2,_,'nnp')],
     [item(_,['endsent','md',':'])]).
rule(seminar, speaker, 4, 0,
     [item(_,'nn'),item(['employers',':'],_),item('endsent','endsent')],
     [item(_,'nnp'),item(_,'nnp')],[item(['from','and'],_),
     item(_,['dt','nnp'])]).
rule(seminar, speaker, 6, 0,
     [],[item(_,'nnp'),item(['carpenter','greiner','coleman','vasulka',
     'mccall'],'nnp')],[]).
rule(seminar, speaker, 3, 0,
     [item([':','by'],_)],[item(['nobuhisa','professor'],'nnp'),list(2,_,'nnp')],
     [item(_,['endsent','in'])]).
rule(seminar, speaker, 3, 0,
     [item(_,['nnp','nn']),item(_,['endsent',','])],[list(2,_,'nnp'),
     item(['curlee','katz'],'nnp')],[]).
rule(seminar, speaker, 6, 0,
     [item(_,['vbz','endsent'])],[item(['mike','toshiaki','terra'],'nnp'),
     item(_,'nnp')],[]).
rule(seminar, speaker, 4, 0,
     [],[item(['alan','mr.','toshinari'],'nnp'),item(_,'nnp')],
     [item(_,[',','md','endsent'])]).
rule(seminar, speaker, 5, 0,
     [item(_,['cd','endsent'])],[item(_,'nnp'),item(['john','c.'],'nnp'),
     item(_,'nnp')],[]).
rule(seminar, speaker, 3, 0,
     [],[item(_,'nnp'),item(['hirschberg','cohen'],'nnp')],[]).
rule(seminar, speaker, 5, 0,
     [],[item(['marc','tom'],'nnp'),list(2,_,'nnp')],[item('endsent','endsent')]).
rule(seminar, speaker, 4, 0,
     [item('endsent','endsent')],[item('professor','nnp'),item(_,'nnp'),item(_,'nnp')],
     [item(_,[',','endsent'])]).
rule(seminar, speaker, 4, 0,
     [],[item(_,'nnp'),item(['manfred','s.'],'nnp'),list(3,_,[',','nnp'])],
     [item('endsent','endsent')]).
rule(seminar, speaker, 5, 0,
     [],[item(['mike','steve'],'nnp'),item(_,'nnp')],[item(_,['nnps','md'])]).
rule(seminar, speaker, 6, 0,
     [item(':',':')],[item(['dr.','dinesh'],'nnp'),item(_,'nnp'),item(_,'nnp')],[]).
rule(seminar, speaker, 4, 0,
     [item(_,['endsent',':'])],[list(2,_,'nnp'),item(['cusumano','jacob'],'nnp')],[]).
rule(seminar, speaker, 3, 0,
     [item(_,[':','nnp'])],[item(_,'nnp'),item(['simmons','gary'],'nnp')],[]).
rule(seminar, location, 29, 0,
     [item(['place','00'],_),item(_,[':','endsent'])],[list(3,_,['nnp','cd'])],
     [item('endsent','endsent'),item(_,['nnp','jj','nn'])]).
rule(seminar, location, 6, 0,
     [item(_,['in','nn']),item(_,['dt','in'])],[list(3,_,['.','(','nnp','pos','cd']),
     item(['room','hall',')'],_),item(',',','),list(2,_,['vbg','nnp','dt']),
     item(_,['cd','nn'])],[]).
rule(seminar, location, 3, 0,
     [item(_,['in','cd']),item(_,_)],[list(2,_,['jj','nnp']),item(['hall','conference'],_),
     item(_,['cd','nn'])],[item(_,['cd','nn'])]).
rule(seminar, location, 3, 0,
     [],[item(['sei','1295'],_),item(_,'nnp'),item(_,_),list(2,_,_),item(_,['nn','nnp'])],
     [item(_,['.','sym'])]).
rule(seminar, location, 7, 0,
     [item(['endsent','in','the'],_)],[item(['wean','room','adamson'],_),item(_,_)],
     [item(_,[':','.'])]).
rule(seminar, location, 6, 0,
     [item(['30','tuesday'],_),item(_,_),item(['endsent','april'],_),
     item(['place','27'],['nn','cd']),item([':',','],_)],[list(7,_,_)],
     [item(['duration','endsent'],_),list(2,_,_),item(_,['cd','dt']),item(_,_)]).
rule(seminar, location, 4, 0,
     [item([':','role'],_),item(['00','of'],_),item(_,_)],[item(_,['cd','nnp']),
     item(_,['nnp','cd'])],[item(_,_),item(_,[':','nnp']),item(_,_),item(_,'nnp')]).
rule(seminar, location, 7, 0,
     [],[item(['faculty','doherty','baker'],_),list(3,_,_),
     item(['skibo','2315','355'],_)],[]).
rule(seminar, location, 5, 0,
     [],[item(_,_),item(['261','floor','conference'],_),item(_,_),item(_,_),
     item(_,['in','nnp']),item(_,_),item(_,['jj','nnp']),item(_,['nn','nnp'])],[]).
rule(seminar, location, 15, 0,
     [],[item(_,'nnp'),item(['2224','hall'],_),list(2,_,_),item(['edrc','100','wing'],_)],
     []).
rule(seminar, location, 3, 0,
     [item(['pm','in'],_),item(_,['endsent','dt'])],[item(_,'nnp'),list(2,_,_),
     item('(','('),item(_,'nnp'),list(2,[')','125','109'],_)],[item(_,['sym','.']),
     item('endsent','endsent')]).
rule(seminar, location, 37, 0,
     [],[item(['gsia','wean','warner','scaife'],'nnp'),item(_,'nnp'),
     item(_,['cd','nnp'])],[]).
rule(seminar, location, 17, 0,
     [],[item(['ucc','wean','hbh'],'nnp'),item(_,'cd')],[]).
rule(seminar, location, 16, 0,
     [item(_,['cd','nn']),item(_,_)],[item(['316','adamson'],['cd','nnp']),item(_,_),
     list(2,_,_),item(_,[',','nnp']),item(_,['nnp','cd'])],[item(['carnegie','endsent'],_)]).
rule(seminar, location, 5, 0,
     [],[item(['room','basement','hamburg'],_),list(3,_,_),
     item(['hbh','library','auditorium'],'nnp')],[]).
rule(seminar, location, 3, 0,
     [item(_,[',','in'])],[list(2,_,['pos','nn','nnp']),item(['2110','office'],_)],[]).
rule(seminar, location, 3, 0,
     [item(_,['nnp','nn']),item(_,_)],[item(_,['nnp','cd']),list(2,_,'nnp'),
     item(['auditorium','tower'],_)],[item(_,[':','.'])]).
rule(seminar, location, 3, 0,
     [],[item(['hh','skibo'],'nnp'),item(_,'nnp')],[]).
rule(seminar, location, 3, 0,
     [item(_,['cd','nnp']),item(_,['in','endsent']),list(2,[':','the','place'],_)],
     [item(['doherty','wherret'],'nnp'),list(3,_,['in','cd','nnp'])],
     [item(_,['dt','endsent'])]).
rule(seminar, location, 4, 0,
     [item(['endsent','location'],_),item(['in',':'],_)],[list(7,_,_)],
     [item(_,[',','endsent']),item(['starting','robert'],_)]).
rule(seminar, location, 5, 0,
     [item(_,['in','cd']),list(2,_,_)],[item(['romm','carnegie'],_),list(2,_,_),
     item(',',','),item(_,'nnp'),list(2,_,'nnp'),item(['center','hall'],'nnp')],[]).
rule(seminar, location, 12, 0,
     [item([':','use'],_)],[item(_,_),item(['hall','2110'],_),item(_,_),
     item(_,[':','nnp']),item(_,_)],[item(_,['endsent','('])]).
rule(seminar, location, 3, 0,
     [],[item(['la','epp'],'nnp'),item(_,_),item(_,_),item(',',','),item(_,_),
     list(2,_,_),item(_,['nnp',':']),item(_,['.','cd'])],[]).
rule(seminar, location, 18, 0,
     [],[item(_,'nnp'),item(['2315','wing'],['cd','nnp']),list(3,_,['in',',','nnp']),
     item('hall','nnp')],[]).
rule(seminar, location, 18, 0,
     [list(2,_,_),item(_,[':',','])],[item(_,'nnp'),item(_,'nnp'),item(_,_),item(_,_),
     item(_,'nnp'),item(['campus','hall'],'nnp')],[]).
rule(seminar, location, 3, 0,
     [item(_,[':','dt'])],[list(2,_,'nnp'),item(['3505','auditorium'],['cd','nn'])],[]).
rule(seminar, location, 25, 0,
     [item(_,['in','endsent'])],[item(_,'nnp'),item(['hall','1004'],_),item(_,['cd','nnp'])],
     [item(_,['.','endsent'])]).
rule(seminar, location, 3, 0,
     [],[item(['porter','ms'],'nnp'),list(2,_,_),item(_,['cd','nnp'])],[item(',',',')]).
rule(seminar, location, 4, 0,
     [],[item('ph','nnp'),item(_,_)],[]).
rule(seminar, location, 3, 0,
     [],[item(['1001','dh'],_),item(_,['nnp','cd'])],[]).
rule(seminar, location, 38, 0,
     [],[item(_,['nnp','cd']),item(['hall','wean'],'nnp'),item(_,['cd','nnp'])],
     [item('endsent','endsent')]).
rule(seminar, location, 34, 0,
     [],[item(['weh','hh'],'nnp'),item(_,'cd')],[]).
rule(seminar, etime, 126, 3,
     [item(['to','-','until'],_)],[list(3,_,_),item(['p.m.','pm','00pm','00'],_)],
     [item(_,['dt',',','endsent','in'])]).
rule(seminar, etime, 5, 0,
     [item('-',':')],[item(['200pm','11am','5pm','1430'],_)],[]).
rule(seminar, etime, 13, 0,
     [item(['00','9am'],'cd'),item('-',':')],[list(3,_,[':','cd'])],
     [item(_,[',','endsent'])]).
rule(seminar, etime, 139, 0,
     [item(['-','until'],_)],[item(_,_),item(':',':'),item(_,'cd'),
     list(1,['a.m.','p.m.','am','pm'],_)],
[item(_,['prp','endsent'])]).
rule(seminar, etime, 84, 4,
     [item('-',':')],[item(['5','12','1'],_),item(':',':'),list(2,_,['nn','cd'])],
     [item(_,['.','endsent','in'])]).
rule(seminar, etime, 10, 0,
     [item(['to','-'],_)],[item(['5','2'],_),list(2,_,['cd','nn',':'])],
     [item(_,['in','nnp','('])]).
rule(seminar, etime, 4, 0,
     [item(['around','to'],_)],[item(['5','11'],_),item(':',':'),item(_,'cd')],[]).
```

B.3 Corporate Acquisitions

The following rules were learned from 300 examples during the experiments in Section 4.8 using Rapier with words and part-of-speech tags.

```
rule(acquisition, status, 20, 0,
     [item(_,['vbd','prp','vbn','to'])],[item(['completed','acquired','terminated',
     'reconsider'],_)],[item(['and','the','because','its'],_)]).
rule(acquisition, status, 3, 0,
     [item(['taft','inc'],'nnp'),list(2,_,_)],[item(['requested','entered'],_),
     list(2,_,_),item(['response','agreement'],'nn')],[item(_,_),item(_,['vb','nnp'])]).
rule(acquisition, status, 22, 1,
     [list(2,_,_)],[item(['rejecting','terminated','completed'],_)],[item(_,_),
     item(_,['nn','jj','rb'])]).
rule(acquisition, status, 5, 0,
     [item('it','prp')],[item(['acquired','sold'],'vbd')],[item(_,['dt','rb']),
     list(2,['the','7.7','all'],_),item(_,['nn','nns'])]).
rule(acquisition, status, 4, 0,
     [item(['trust','will','inc'],_)],[item(['sold','offer','acquired'],_)],[]).
rule(acquisition, status, 15, 0,
     [],[item(['tentiative','ended','definitive','received'],_),
     item(['talks','agreement','approvcal'],_)],[item(_,['to','in'])]).
rule(acquisition, status, 6, 0,
     [],[item(_,['vbn','vbz']),item(['to','no'],_),item(['buy','limit','bid'],_)],
     [item(_,['dt','jj','in'])]).
rule(acquisition, status, 5, 0,
     [item(_,[',','nn',':']),item(_,['in','nnp']),item([',','norstar','corp'],_),
     item(_,_),item(_,['vbn','vbp','prp'])],[item(['purchased','agreed'],_)],
     [item(['by','to'],_)]).
rule(acquisition, status, 3, 0,
     [list(4,_,_)],[item(['revision','proposed','rejecting'],_)],
     [item(['to','merger','this'],_),list(2,_,_),item(_,['dt','nnp','pos'])]).
rule(acquisition, status, 3, 0,
     [item(_,[',','vbd']),item(['said','it'],_),item(['it','has'],_)],
     [list(3,['approvals','regulatory','acquired','received'],_)],[item(['a','for'],_),
     item(_,['nn','prp$'])]).
rule(acquisition, status, 6, 0,
     [],[item(['not','agreed','letter','decided'],_),item(_,_),
     item(['discussions','buy','intent','buying'],_)],[item(['with','the','charmglow'],_)]).
rule(acquisition, status, 3, 0,
     [item(_,['prp','vbp'])],[list(2,_,['dt','rb','vbd']),item(['true','agreement'],_)],
     [item(_,['to','.']),item(['sell','endsent'],_)]).
rule(acquisition, status, 9, 0,
     [item(_,['in','prp']),item(['the','has'],_)],[item(['agreed','agreement'],['vbn','nn']),
     list(2,_,_),item(['principle','acquire','sell'],_)],[list(2,_,_),item(_,['dt','nnp'])]).
rule(acquisition, status, 3, 0,
     [item(_,['nnp',',']),list(2,_,_),item(['agreed','reached'],'vbd'),item(['to','an'],_)],
     [item(['agreement','move'],_),list(3,_,['nn','dt','to','in']),item(_,['vb','nn'])],
     [list(2,_,_),item(_,_),item(_,_),item(_,_),item(_,'nnp')]).
rule(acquisition, status, 8, 0,
     [item(_,'nnp'),item(_,_),item(_,['prp','vbz'])],[item(['agreed','no'],_),
     list(3,['an','in','to','interest'],_),item(['sell','acquisition'],_)],[]).
rule(acquisition, status, 3, 0,
     [item('has','vbz')],[item(['bought','acquired'],'vbn')],[item(_,['nnp','cd']),
     list(2,['600','-',','],_),item(_,['nnp','nns'])]).
rule(acquisition, status, 9, 0,
     [],[item(['seeking','agreed','plans','ended'],_),item(_,_),
     item(['merger','principle','complete','agreement'],_)],[item(_,_),
     item(['kappa','acquire','integration','endsent'],_)]).
rule(acquisition, status, 4, 0,
     [item(_,'nnp'),item(['said','corp'],['vbd','nnp']),item(['it','}'],_)],
     [item(['agreed','will'],_),list(3,['tender','a','to','make'],_),item(_,['vb','nn'])],
     [item(_,_),item(_,['dt','jj']),item(_,['cc','nnp'])]).
rule(acquisition, status, 8, 0,
     [item(_,['prp','vbp']),item(_,_)],[item(['discussing','agreed'],_),item(_,_),
     item(_,['nn','vb'])],[list(2,_,_),item(_,['dt','nn'])]).
rule(acquisition, status, 11, 0,
     [item(_,'nnp'),item(_,'vbd'),item(['still','it'],_)],[item(['under','agreed'],_),
     list(2,_,_)],[item(['and','consolidated','its','{'],_)]).
rule(acquisition, status, 3, 0,
     [item(_,['in','vbd']),item(_,['dt','prp'])],[item(['offer','sold'],_)],
     [item(_,['prp','prp$']),list(2,_,_),item(_,['nn','nnp'])]).
rule(acquisition, status, 3, 0,
     [],[item(['agreement','received'],_),list(2,_,_),item(['principle','proposal'],'nn')],
     [item(_,_),item(['j.p.','acquire'],_)]).
rule(acquisition, status, 3, 0,
     [item(_,['nnp','vbd']),item(_,_),item(['it','reached'],_),item(['has','an'],_)],
     [list(3,['principle','in','acquired','agreement'],_)],[item(['a','to'],_),
     list(2,['seven','90','buy'],_),item(_,['nn','nnp'])]).
rule(acquisition, status, 3, 0,
     [item(_,['vbd','nnp']),list(2,_,_)],[item(['agreement','sold'],['nn','vbd'])],
     [item(_,['md','cd'])]).
rule(acquisition, status, 10, 0,
     [],[item(['management','agreed'],_),item(_,['nn','in']),item(_,'nn')],[list(2,_,_),
     item(_,['dt','nnp'])]).
rule(acquisition, status, 4, 0,
     [item(_,['vbd',')']),item(_,_),item(_,_)],[item(['started','entered'],_),
     item(_,_),item(_,_),item(_,'nn')],[]).
rule(acquisition, status, 3, 0,
     [item(['inc','said'],_),item(_,_),item(_,['prp','vbz']),item(['has','been'],_)],
     [list(3,_,['vb','to','vbn'])],[item(['{','by'],_),item(_,_),item(_,['nnp','nn'])]).
rule(acquisition, status, 3, 0,
     [item(_,[')','nnp']),item(_,['nn','vbd']),item(['had','it'],_)],[list(2,_,_),
     item(['acquire','sell'],'vb')],[item(_,['(','dt']),item(_,['nnp','cd'])]).
rule(acquisition, status, 17, 0,
     [list(2,_,_)],[item(['sold','completed'],_)],[item('the','dt')]).
rule(acquisition, status, 6, 0,
     [],[item(['request','exercise','entered'],['nn','vb','vbn']),list(4,_,_),
     item(['proposals','acquire','agreement'],_)],[]).
rule(acquisition, status, 7, 0,
     [item(_,['nns','nnp']),item(_,_),item(_,['endsent','vbd']),item(['the','it'],_)],
     [item(['bid','acquired'],_)],[list(2,_,_),list(2,_,_),item(_,['nnp','jj'])]).
rule(acquisition, status, 6, 0,
     [item(_,['vbd','prp'])],[item(['agreed','talks'],_),list(2,_,_),
     item(['buy','place'],_)],[item(_,['wdt','dt'])]).
rule(acquisition, status, 4, 0,
     [list(2,['have','it','they'],_)],[item(['acquired','bought'],_)],[item(_,'cd'),
     item(',',','),item('000','cd')]).
rule(acquisition, status, 3, 0,
     [],[item(['acquired','bought'],'vbd')],[item(['an','two'],_),item(_,['nn','jjr'])]).
rule(acquisition, status, 3, 0,
     [item(['corp','co'],'nnp'),item(_,['vbd',',']),item(_,['prp','vbd']),
     item(['has','it'],_)],[item(['acquired','agreed'],_)],[item(_,['dt','to']),
     item(_,_),list(2,_,_),item(_,['in','jj'])]).
rule(acquisition, status, 3, 0,
     [item(['merger','shareholders'],_)],[item('approved','vbd')],[]).
rule(acquisition, status, 5, 0,
     [list(2,['is','a','it'],_)],[item(['seeking','letter'],_),item(_,_),
     item(['sell','intent'],_)],[item(_,['prp$','to'])]).
rule(acquisition, status, 3, 0,
     [list(2,['has','the',','],_)],[item(['purchased','offer'],_)],[item(_,['cd','vbz']),
     item(_,[',','jj'])]).
```

```
rule(acquisition, status, 4, 0,
     [list(2,_,_)],[item([’acquired’,’sold’],_)],[item([’the’,’78’],_),
     list(2,_,_),item(_,’nns’)]).
rule(acquisition, status, 7, 0,
     [item(_,[’fw’,’nnp’]),item(_,_),item(_,_),item(_,[’nns’,’nnp’]),item(_,_),item(_,_),
     item([’in’,’it’],_),item(_,[’prp$’,’vbz’])],[item([’proposed’,’completed’],’vbn’)],
     []).
rule(acquisition, status, 7, 0,
     [item(_,[’nnp’,’vbd’]),item(_,[’vbz’,’prp’])],[list(2,_,_),
     item([’sell’,’acquire’],’vb’)],[item([’all’,’its’],[’dt’,’prp$’]),item(_,’nnp’)]).
rule(acquisition, status, 4, 0,
     [],[item([’terminated’,’continues’],_),list(2,_,[’vb’,’dt’,’to’]),
     list(2,[’sale’,’agreement’,’for’],_)],[item(_,[’in’,’"’])]).
rule(acquisition, status, 7, 0,
     [item([’been’,’has’],_)],[item([’terminated’,’completed’],’vbn’)],[item(_,_),
     item(_,[’jj’,’rb’])]).
rule(acquisition, status, 5, 0,
     [list(2,_,_),item(_,[’vbz’,’vbp’])],[item([’acquired’,’ended’],’vbn’)],[list(2,_,_),
     item(_,[’dt’,’nnp’]),item(_,[’in’,’nnp’])]).
rule(acquisition, status, 5, 0,
     [item(_,[’nn’,’vbd’]),item([’has’,’it’],_)],[list(2,[’negotiations’,’completed’,
     ’started’],_)],[item([’with’,’its’],_)]).
rule(acquisition, status, 11, 0,
     [],[item([’agreed’,’found’],’vbn’),list(2,_,_),item([’principle’,’buyer’],’nn’)],
     []).
rule(acquisition, status, 11, 0,
     [item(_,[’prp’,’rb’])],[item([’completed’,’rejected’],’vbd’)],[item(_,’dt’)]).
rule(acquisition, sellercode, 12, 0,
     [item(’{’,’(’)],[item(_,’nnp’)],[item(’}’,’)’),item([’to’,’explores’],_),
     item([’complete’,’sell’,’unit’],_),item(_,[’nn’,’dt’,’nnp’])]).
rule(acquisition, sellercode, 15, 0,
     [],[item([’lrho.l’,’hepc.l’,’atco.o’,’ckb’,’hnsn.l’,’bac’,’crrs’,’ftr’,’kb’,’bor’,
     ’tae’,’gm’,’han’],_)],[]).
rule(acquisition, sellercode, 9, 0,
     [item(’{’,’(’)],[item(_,’nnp’)],[item(’}’,’)’),item([’for’,’,’,’sells’,’completes’],_),
     item([’235’,’who’,’sale’,’foam’,’30’],_)]).
rule(acquisition, sellercode, 6, 0,
     [],[item([’bmy’,’slb’,’bnl’,’gy’,’esk’],’nnp’)],[item(’}’,’)’),
     item(_,[’.’,’vbz’,’nn’])]).
rule(acquisition, sellercode, 7, 0,
     [item(’{’,’(’)],[item(_,’nnp’)],[item(’}’,’)’),item([’signs’,’unit’],_),
     item([’pact’,’endsent’],_),item(_,[’in’,’nnp’]),item(_,[’nnp’,’,’]),item(_,_),
     item(_,[’nnp’,’.’])]).
rule(acquisition, sellercode, 3, 0,
     [item([’’’s’,’dome’],[’pos’,’nnp’]),item(’{’,’(’)],[item(_,’nnp’)],[item(’}’,’)’),
     item(_,[’nn’,’nnp’]),item(_,[’nns’,’nnp’]),list(2,_,_),item(_,[’(’,’nn’]),
     item(_,[’in’,’nnp’])]).
rule(acquisition, sellercode, 15, 0,
     [item(’{’,’(’)],[item(_,’nnp’)],[item(’}’,’)’),item([’sells’,’selling’,’south’],_),
     list(2,_,_),item(_,[’endsent’,’nn’])]).
rule(acquisition, sellercode, 8, 0,
     [item(’{’,’(’)],[item(_,’nnp’)],[item(’}’,’)’),item([’sells’,’seeks’,’ends’],’vbz’),
     item(_,_),item(_,[’nnp’,’vb’,’to’])]).
rule(acquisition, sellercode, 3, 0,
     [item(_,’nnp’),item([’corp’,’healthcare’],’nnp’),item(’{’,’(’)],[item(_,’nnp’)],
     [item(’}’,’)’),item(_,[’.’,’to’]),list(2,_,_),item(_,_),item(_,[’nnp’,’endsent’])]).
rule(acquisition, sellercode, 3, 0,
     [item(_,’nnp’),item(_,’nnp’),item(’{’,’(’)],[item(_,_)],[item(’}’,’)’),
     item([’agrees’,’completes’],’vbz’),list(2,_,_),item([’sell’,’sale’],_)]).
rule(acquisition, sellercode, 3, 0,
     [item([’harcourt’,’buy’],_),item(_,’nnp’),item(’{’,’(’)],[item(_,’nnp’)],
     [item(’}’,’)’),item(’unit’,’nn’)]).
```

```
rule(acquisition, sellercode, 6, 0,
     [item(_,_),item(_,_),item('{','(')],[item(_,'nnp')],[item('}',')'),
     item(_,[',','to']),item(['usair','sell'],_),item(_,['(','nnp'])]).
rule(acquisition, sellercode, 3, 0,
     [item(_,[',','nnps']),item(['but','{'],_)],[item(_,['nnp','nns'])],
     [item(_,['vbz',')']),item(_,_),item(_,['in','vb'])]).
rule(acquisition, sellerabr, 3, 0,
     [item(['houston','denver'],'nnp'),item(',',','),item(['march','april'],'nnp'),
     item(['4','1'],'cd'),item('-',':')],[list(2,_,'nnp')],[item(['inc','corp'],'nnp'),
     item('said','vbd'),item('it','prp')]).
rule(acquisition, sellerabr, 3, 0,
     [item(_,['nn','nnp']),list(2,_,_),item(['.','subsidiary'],_),
     item(_,['endsent','in'])],[item(_,'nnp'),item(_,'nnp')],[item(['said','corp'],_),
     item(_,_),item(_,['nns','nnp'])]).
rule(acquisition, sellerabr, 3, 0,
     [item(['1','2'],'cd'),item('-',':')],[item(_,'nnp')],[item(['inc','corp'],'nnp'),
     item('said','vbd'),item(_,_),item(_,['nnp','vbz'])]).
rule(acquisition, sellerabr, 4, 0,
     [item(['acquiring','from','-'],_)],[item(['trimac','quaker','butler'],'nnp')],
     []).
rule(acquisition, sellerabr, 3, 0,
     [item(_,['rb','nn']),item(_,_),item(['plants',','],_),list(2,_,_),
     item(_,['nn','vbn']),item(['for','by'],'in')],[item(_,'nnp')],[item(_,['.',',']),
     item(_,_),item(_,['nnp','rb'])]).
rule(acquisition, sellerabr, 3, 0,
     [item(_,['in','cd']),item(_,['(',':'])],[list(2,_,'nnp')],[item(['mines','corp'],_),
     item(['inc','said'],_),item(_,[')','prp']),item(['in','completed'],_),
     item(_,['jj','dt']),item(['mining','sale'],'nn')]).
rule(acquisition, sellerabr, 4, 0,
     [item(_,['nnp','nn']),item(_,_),item(_,_),list(2,['indentify','march','the',
     'pct'],_),item(['2','buyer','owned'],_),item(_,['in',':'])],[item(_,'nnp')],
     [item(['corp','''s','mining'],_),list(3,_,_),item(_,[',','dt','nns'])]).
rule(acquisition, sellerabr, 4, 0,
     [item(_,['nnp','vbd','jj']),item(_,_),item(['march','this','an'],_),
     item(['9','transaction','issue'],_),item(_,[':',',','to'])],[item(_,'nnp')],
     [item(_,['nnp','rb',',']),list(3,_,_),item(_,['jj','in','dt'])]).
rule(acquisition, sellerabr, 3, 0,
     [],[item(['kaufman','allied'],_),item(_,_),item(['board','signal'],_)],[]).
rule(acquisition, sellerabr, 3, 0,
     [item(['pct','3'],_),item(['of','-'],_)],[item(_,'nnp'),item(_,'nnp')],
     [item(_,['pos','nnp']),item(_,['jj','vbd']),item(_,_),item(['of','agreed'],_)]).
rule(acquisition, sellerabr, 6, 0,
     [],[item(['hepworth','chesapeake','chevron','allied','quest'],'nnp')],
     [item(_,['vbz','nnp','pos','vbd'])]).
rule(acquisition, sellerabr, 3, 0,
     [item(['who','march'],_),item(_,_),item(['hearing','-'],_)],[list(2,_,'nnp')],
     [item(['steel','ltd'],'nnp'),item(_,['pos','vbd']),list(2,_,_),
     item(_,['cd','vbn'])]).
rule(acquisition, sellerabr, 3, 0,
     [item(_,['vbg','jj']),list(2,_,_),item(['buy','endsent'],_)],[item(_,'nnp'),
     item(_,['nnp','nnps'])],[item(_,['nnp','vbn']),item(_,_),
     list(2,['of','endsent','purchase'],_),item(_,['dt','nnp'])]).
rule(acquisition, sellerabr, 4, 0,
     [],[item(['borg','texas','johnson'],'nnp'),
     list(2,['-','american','warner','products'],_)],[item(['corp','{','to'],_)]).
rule(acquisition, sellerabr, 3, 0,
     [item(['25','3'],'cd'),item('-',':')],[item(_,'nnp'),list(3,_,['nnp','nn',':'])],
     [item(['corp','corfp'],'nnp'),item('said','vbd')]).
rule(acquisition, sellerabr, 3, 0,
     [item(_,['nnp','jj']),item(_,_),item(['april','of'],_),item(_,['cd','rb']),
     item(_,[':','vbn'])],[list(2,_,'nnp')],[item(['inc','10'],_),item(['said','-'],_),
     item(_,['prp','cd']),list(2,_,_),item(_,_),item(['the','debentures'],_)]).
```

```
rule(acquisition, sellerabr, 3, 0,
     [item(_,['md','endsent']),list(2,_,_),item(_,['nnp','cd']),item(_,_),
     item(['-','000'],_)],[item(_,'nnp')],[item(['international','shares'],_),
     item(_,['nnp','in']),item(_,['vbd','nnp']),item(_,['prp','nnp']),
     item(_,['vbd','nnp'])]).
rule(acquisition, sellerabr, 3, 0,
     [item(_,['vbn','vbp']),item(_,_),item(_,['nnp',',']),item(['hubert','"'],_)],
     [item(_,'nnp')],[item(_,_),item(_,['nn','.'])]).
rule(acquisition, sellerabr, 3, 0,
     [item(['cleveland','toronto'],'nnp'),item(',',','),item('march','nnp'),
     item(['13','6'],'cd'),item('-',':')],[list(2,_,['nnp','vbg']),item(_,'nnp')],
     [item(['enterprises','ltd'],_)]).
rule(acquisition, sellerabr, 3, 0,
     [item(['stevenson','.'],_),item(_,['vbd','endsent'])],[list(2,_,'nnp')],
     [item(['will','said'],_),item(['continue','the'],_),item(['to','sale'],_),
     item(_,['vb','vbz'])]).
rule(acquisition, sellerabr, 3, 0,
     [item(['{','buys'],_)],[item(_,'nnp'),item(_,'nnp')],[item(['inc','{'],_),
     item(_,[')','nnp']),item(['for','}'],_),list(2,_,_),item(_,'nn')]).
rule(acquisition, sellerabr, 3, 0,
     [item(['hands','york'],_),list(2,_,_),item(_,['endsent','cd']),item(_,['dt',':'])],
     [list(2,_,'nnp')],[item(['sources','corp'],_),item(_,'vbd'),item(['no','it'],_),
    item(_,['nns','vbz'])]).
rule(acquisition, sellerabr, 3, 0,
     [item(_,'nnp'),item(',',','),item(_,'nnp'),item(['2','6'],'cd'),item('-',':')],
     [list(2,_,'nnp')],[item(['corp','ltd'],'nnp'),item(_,_),item(_,['prp','cd']),
     list(2,_,_),item(_,['dt','in'])]).
rule(acquisition, sellerabr, 3, 0,
     [item(['the','buy'],_)],[item(_,'nnp')],[item(['said','''s'],_),
     item(_,['.','nnp'])]).
rule(acquisition, sellerabr, 3, 0,
     [item(['oct','substantial'],_),item(['19','developments'],_),
     item(['-','affecting'],_)],[item(_,'nnp'),item(_,'nnp')],[item(_,['nnp',',']),
     item(_,_),item(_,['prp$','nn'])]).
rule(acquisition, sellerabr, 3, 0,
     [item(_,['wrb','in'])],[item(['hanson','cpc'],'nnp')],[item(_,['vbd','nnp'])]).
rule(acquisition, sellerabr, 3, 0,
     [item(_,['nnp',')']),item('to','to'),item('buy','vb')],[item(_,'nnp')],
     [item(['division','{'],_)]).
rule(acquisition, sellerabr, 3, 0,
     [item(_,['dt','vbd']),list(2,['endsent','local','.'],_),
     item(['annesley','subsidiary'],_),item(_,['rb','in'])],[item(_,'nnp')],
     [item(['petroleum','corp'],'nnp')]).
rule(acquisition, seller, 6, 0,
     [item(['subsidiary',',','3'],_),item(['of','"','-'],_)],[list(2,_,'nnp'),
     item(['corp','industries','co'],_)],[item(_,_),item(_,['in','nnp','prp'])]).
rule(acquisition, seller, 4, 0,
     [item(['consumer','endsent','akron'],_),list(3,_,_),item(_,_),
     item(['accounts','20','1'],_),item(['from','-'],_)],[list(2,_,'nnp'),
     item(['co','inc','federal'],'nnp')],[item(_,['(','vbd']),
     list(4,_,_),item(_,[')','nnp','dt'])]).
rule(acquisition, seller, 4, 0,
     [item(['4','2','19'],'cd'),item('-',':')],[item(_,'nnp'),
     item(['stores','sandwiches','industries'],_),item('inc','nnp')],[]).
rule(acquisition, seller, 5, 0,
     [item(['march','houston','chief'],_),item(_,_),item(['-','from','officer'],_)],
     [list(2,_,_),item(['corfp','inc','cohen','l.p.'],'nnp')],[item(_,_),
     item(_,['dt','cd','prp']),item(_,_),item(_,_),item(['advisory','a','.','shares'],_),
     item(_,['nn','jj','endsent','in'])]).
rule(acquisition, seller, 7, 0,
     [item(_,[':','vb','('])],[item(['borealis','riedel','general','allied','deltec',
     'kaufman'],'nnp'),list(2,_,_),item(['sa','corp','inc','ltd'],'nnp')],[]).
```

```
rule(acquisition, seller, 3, 0,
     [list(3,_,_)],[item(_,'nnp'),item(['holdings','l.','development'],'nnp'),
     item(_,'nnp')],[item(_,[')',',','vbd']),list(3,_,_),item(_,'nnp')]).
rule(acquisition, seller, 3, 0,
     [item(['chicago','york'],'nnp'),item(',',','),item(_,'nnp'),item(['2','26'],'cd'),
     item('-',':')],[item(_,'nnp'),item('corp','nnp')],[item('said','vbd'),
     item('it','prp'),item(_,['vbd','vbz'])]).
rule(acquisition, seller, 4, 0,
     [item(['from','inc'],_),item(_,['(','pos'])],[list(4,_,_),item('inc','nnp')],
     [item(_,[')','('])]).
rule(acquisition, seller, 8, 0,
     [],[item(['allied','owens','cpc','cvs','creditors'],_),list(3,_,_),
     item(['corp','inc','facilities'],_)],[]).
rule(acquisition, seller, 3, 0,
     [item(['april','{'],_),item(_,_),item(_,[':',')'])],[list(2,_,['jj','nnp']),
     item(['chemical','holdings'],'nnp'),item(_,'nnp')],[item(_,_),
     list(2,['undisclosed','hepc.l','for'],_),item(_,_),item(_,['vbd','.'])]).
rule(acquisition, seller, 5, 0,
     [item(_,['vbn',':','('])],[item(['rhone','hanson','borg','canadian'],'nnp'),
     list(2,_,_),item(['chimie','plc','corp','ltd'],'nnp')],[]).
rule(acquisition, seller, 4, 0,
     [item(['magnetics','march'],_),item(_,['nn','cd']),item(['of','-'],_)],
     [item(_,'nnp'),list(2,['system','mccormack','industries','resources','bank'],'nnp'),
     item('inc','nnp')],[item(_,[',','vbd']),item(_,_),item(_,['nn','vbz']),
     item(['found','of','completed'],_)]).
rule(acquisition, seller, 3, 0,
     [item(['chairman','from'],_)],[list(2,_,'nnp'),item(['brinkman','corp'],'nnp')],
     [item(_,['cc',','])]).
rule(acquisition, seller, 3, 0,
     [item(['air','march'],'nnp'),item(['corp','17'],_),item(_,['pos',':'])],
     [list(2,_,'nnp'),item(['airlines','corp'],_)],[item('said','vbd'),
     item(_,['prp','jj'])]).
rule(acquisition, seller, 3, 0,
     [item(_,['in','nnp']),item(_,_),item(['capital','march'],_),item(_,['nn','cd']),
     item(_,['in',':'])],[item(_,'nnp'),item('inc','nnp')],[item(_,['pos','vbd']),
     item(_,['rb','prp']),item(_,_),list(2,['d','completed','subsidiary'],_),
     item(_,['dt','cc'])]).
rule(acquisition, seller, 3, 0,
     [list(2,_,_)],[item(['nordbanken','gencorp'],'nnp')],[item(_,[')','vbd'])]).
rule(acquisition, seller, 3, 0,
    [item('9','cd'),item('-',':')],[list(3,_,'nnp'),item(['partners','inc'],_)],
     [item(_,'vbd'),item(_,_),item(_,['vbd','nnps']),item(_,_),item(_,['nn','nnp'])]).
rule(acquisition, seller, 3, 0,
     [item(_,['vbn','vb']),item(_,['in','cd'])],[list(2,_,'nnp'),
     item(['hat','corp'],'nnp')],[item(_,[',','nn'])]).
rule(acquisition, seller, 3, 0,
     [item(_,['vb',')']),item('{','(')],[list(5,_,_),item('ltd','nnp')],[]).
rule(acquisition, seller, 4, 0,
     [item(['pct','subsidiary','substantially'],_),item(['of','all'],_)],
     [list(3,_,['cc','nnp']),item(_,'nnp'),item(['usa','corp','associate'],'nnp')],
     [item(_,['(','pos'])]).
rule(acquisition, seller, 4, 0,
     [item(_,['nnp','nn']),item(['-','of'],_),item(_,['vbn','('])],[list(2,_,'nnp'),
     item(['inc','ab'],'nnp')],[item(_,['vbd',')'])]).
rule(acquisition, seller, 3, 0,
     [item(['march','new'],'nnp'),item(_,['cd','nnp']),item(_,[':','nn'])],
     [list(3,_,['nnps','nnp']),item(['edelman','co'],'nnp')],[item(['said','and'],_),
     list(2,_,_),item(['agreed','inc'],_),list(2,_,_),item(_,['nnp','prp$'])]).
rule(acquisition, seller, 3, 0,
     [item(_,['nnp','cc']),item(_,_),item(['april','shield'],_),item(_,['cd',',']),
     item(['-','while'],_)],[list(2,_,'nnp'),item('corp','nnp')],[item(_,['vbd','md']),
     item(_,['prp','vb']),item(_,_),item(_,['to','vb'])]).
```

```
rule(acquisition, seller, 5, 0,
     [item(['edelman','of','18'],_),item(_,['cc','(',':'])],[item(_,'nnp'),
     item(_,'nnp'),item(['inc','co'],'nnp')],[item(_,['(',')','vbd']),
     item(_,['nnp','in','prp'])]).
rule(acquisition, seller, 3, 0,
     [list(2,_,_),item(_,['cc','('])],[item(_,'nnp'),item(['l.','holdings'],'nnp'),
     item(_,'nnp')],[item(_,[',',')']),item(_,_),list(2,_,_),item(_,'nnp')]).
rule(acquisition, seller, 3, 0,
     [item(['-','with'],_)],[list(2,_,'nnp'),item(['city','southern'],'nnp'),item(_,_),
     item(_,'nnp')],[]).
rule(acquisition, seller, 4, 0,
     [item(_,['nnp','vb']),list(2,['the','2','shares','6'],_),item(_,[':','in'])],
     [item(_,_),item(['finance','industries','o'],'nnp'),list(2,_,'nnp')],
     [item(_,['vbd',','])]).
rule(acquisition, seller, 3, 0,
     [item(['.','detroit'],_),item(',',','),item('march','nnp'),item(['24','31'],'cd'),
     item('-',':')],[item(_,'nnp'),list(2,_,'nnp'),item(['inc','corp'],'nnp')],
     [item('said','vbd'),list(2,_,_),item(_,['nnp','to'])]).
rule(acquisition, seller, 3, 0,
     [],[item(['u.s.','allegheny'],'nnp'),item(_,'nnp'),item(_,'nnp')],[item(_,'vbd')]).
rule(acquisition, seller, 4, 0,
     [item(_,['vbn','nnp']),item(['{','from'],_)],[list(2,_,'nnp'),
     item(['inc','co'],'nnp')],[item(_,_),item(_,['.','cd'])]).
rule(acquisition, seller, 3, 0,
     [item(_,['cd','vbn']),item(_,[':','('])],[item(_,'nnp'),list(2,_,['nnp',':']),
     item(['ltd','corp'],'nnp')],[item(_,['pos',')']),item(_,_),list(2,_,_),
     item(_,['in','vbn']),item(_,_),item(_,_),item(_,['nnp','jj'])]).
rule(acquisition, seller, 3, 0,
     [item(_,['nn','nnp']),list(2,['venture','from','joint'],_),item(['from','{'],_)],
     [item(_,'nnp'),list(2,['mines','minerals','gold'],'nnp'),item('inc','nnp')],
     [item(_,_),item(_,['dt','in'])]).
rule(acquisition, seller, 3, 0,
     [item([',','by'],_)],[list(2,_,'nnp'),item(['ltd','myers'],'nnp')],
     [item(_,_),item(_,'nnp')]).
rule(acquisition, seller, 3, 0,
     [item(_,[',','in'])],[list(2,_,_),item(['resources','myers'],'nnp'),item(_,'nnp')],
     [item(_,_),item(_,'nnp')]).
rule(acquisition, seller, 3, 0,
     [item(['from','-'],_)],[list(2,_,'nnp'),item(['resources','energy'],'nnp'),
     item('corp','nnp')],[]).
rule(acquisition, seller, 3, 0,
     [],[item(['bank','kaiser'],'nnp'),item(_,_),item(['america','corp'],'nnp')],
     []).
rule(acquisition, seller, 3, 0,
     [item(['2','portfolio'],_),item(['-','of'],_)],[list(2,_,'nnp'),
     item(['international','credit'],'nnp'),item(_,'nnp')],[]).
rule(acquisition, purchcode, 4, 0,
     [],[item(['abi','u','ldco.o'],'nnp')],[item('}',')'),item(_,[',','vbz','nn']),
     list(2,_,_),item(_,['vbn','nn','endsent'])]).
rule(acquisition, purchcode, 4, 0,
     [item(['ltd','oats'],'nnp'),item('{','(')],[item(_,'nnp')],[item('}',')'),
     item(_,['vbd','vbz']),item(_,['dt','prp','nnp'])]).
rule(acquisition, purchcode, 7, 0,
     [item(_,['jj','nnp']),item('{','(')],[item(_,'nnp')],[item('}',')'),
     item(['affiliation','to','cleared'],_),item(['endsent','acquire','to'],_)]).
rule(acquisition, purchcode, 43, 2,
     [],[item(['kra','rads','brf','mits.t','ma','amkg','atcma','ko','xtr','tjco','bls',
     'vido','mony.o','ppw','tpsi','hrcly.o','aare','cfmi','forf','mmm','jn.to','twa',
     'kdi','lvi','cmco','dci','clg','narr','rtrsy','len','comm.o','ssbk','pkn','csra.s',
     'keyc','mer','jpi'],'nnp')],[]).
rule(acquisition, purchcode, 4, 0,
     [item(['fielder','to','hampshire'],_),item(_,'nnp'),item('{','(')],[item(_,_)],
     [item('}',')'),item(_,['in','endsent','to'])]).
rule(acquisition, purchcode, 4, 0,
     [item(['plc','scientific','alza'],'nnp'),item('{','(')],[item(_,'nnp')],
     [item('}',')'),item(_,['vbd','in','endsent']),list(2,_,_),
     item(_,['to','nn',','])]).
rule(acquisition, purchcode, 6, 0,
     [item(['inc','mortgage','fleet','greyhound'],'nnp'),item('{','(')],[item(_,_)],
     [item('}',')'),item(['outline','buys','said'],_),item(_,['nn','nnp','prp'])]).
rule(acquisition, purchcode, 20, 1,
     [],[item(['darta.o','btek.o','dyr','fwf','anwa.t','pant','asrn.as','lsb','soi',
     'forf','rabt','fsb','nva.a.to','lce','trb','duri','laf'],'nnp')],[]).
rule(acquisition, purchcode, 5, 0,
     [],[item(['spc','func','c','uac'],_)],[item('}',')')]).
rule(acquisition, purchcode, 4, 0,
     [item(['ssmc','plc','henley'],'nnp'),item('{','(')],[item(_,'nnp')],
     [item('}',')'),item(_,['to','vbg','vbz'])]).
rule(acquisition, purchcode, 8, 0,
     [item(_,['nns','nnp']),item('{','(')],[item(_,'nnp')],[item('}',')'),
     item(_,[',','to','vbd']),item(_,_),item(['bank','acquisition','would'],_)]).
rule(acquisition, purchcode, 7, 0,
     [item(_,['nnp','nns']),item('{','(')],[item(_,'nnp')],[item('}',')'),
     item(['sets','acquires','acquisitions'],_),item(_,['nnp','vbd'])]).
rule(acquisition, purchcode, 7, 0,
     [item(_,'nnp'),item('{','(')],[item(_,'nnp')],[item('}',')'),item(['to','said'],_),
     item(['purchase','buy','its'],_),item(_,['nnp','vbp']),item(_,['nnp','nn'])]).
rule(acquisition, purchcode, 5, 0,
     [item(_,_),item(_,['nnp','vb']),item('{','(')],[item(_,_)],[item('}',')'),
     item(['gets','has','says'],'vbz')]).
rule(acquisition, purchcode, 6, 0,
     [item(['plc','monsanto','allwaste','of'],_),item(_,['(','dt'])],[item(_,'nnp')],
     [item(['}','group'],[')','nnp']),item(_,['in','to','cc'])]).
rule(acquisition, purchcode, 7, 0,
     [item(_,'nnp'),item('{','(')],[item(_,'nnp')],[item('}',')'),
     item(_,['vbz','cc']),item(['acquisition','has'],_)]).
rule(acquisition, purchcode, 3, 0,
     [item(_,['nnp',':']),item(['co','tv'],_),item(['{','to'],_)],[item(_,'nnp')],
     [item(['}','inc'],_),item(_,_),item(_,['vbn','nnp'])]).
rule(acquisition, purchcode, 3, 0,
     [item(_,'nnp'),item('{','(')],[item(_,'nnp')],[item('}',')'),
     item(_,[':','md'])]).
rule(acquisition, purchcode, 3, 0,
     [item(['march','industries'],'nnp'),item(['23','inc'],_),item(_,[':','('])],
     [item(_,'nnp')],[item(['group','}'],_),item(_,_),item(_,['vbd','endsent'])]).
rule(acquisition, purchcode, 7, 0,
     [item(_,_),item('{','(')],[item(_,'nnp')],[item('}',')'),
     item(['buys','purchases'],_),item(_,_),item(_,_),item(_,'nnp')]).
rule(acquisition, purchaser, 8, 0,
     [item(['inc','march','25'],_),item(_,['nn',':']),item(_,['to',':','('])],
     [list(5,_,_),item(['corp','inc'],'nnp')],[]).
rule(acquisition, purchaser, 3, 0,
     [item(['with','-'],_)],[item(_,'nnp'),item(['inc','partners'],_)],
     [item(_,['(',',']),item(_,['nnp','dt'])]).
rule(acquisition, purchaser, 5, 0,
     [item(_,[':','('])],[list(2,_,'nnp'),item(_,_),item(['forbes','insurance',
     'west'],'nnp'),item(['group','co','inc'],'nnp')],[]).
rule(acquisition, purchaser, 6, 0,
     [item(['rural','april','oct','pictures'],_),item(_,_),item(_,[':','to'])],
     [item(_,['nnp','dt']),item(['private','group','commercial','management'],_),
     item(_,['nnp','nn'])],[]).
rule(acquisition, purchaser, 3, 0,
     [item(_,['vbd',':'])],[list(3,_,['pos','nnp']),item(['g.','american'],'nnp'),
     item(_,['nnp','nnps'])],[item(_,['vbd','nn'])]).
```

```
rule(acquisition, purchaser, 6, 0,
     [item(['march','24','20'],_),item(_,_),item(_,[':','(','dt'])],[list(2,_,'nnp'),
     item(['systems','aerospace','manhattan'],'nnp'),item(_,'nnp')],[list(2,_,_),
     item(_,['(','vbd','nn'])]).
rule(acquisition, purchaser, 5, 0,
     [item(_,['(','dt'])],[item(_,'nnp'),item(['plc','government'],_)],
     [item(_,[')','vbz']),item(_,['in','vbd','vbn'])]).
rule(acquisition, purchaser, 5, 0,
     [item(['19','to'],_),item(_,[':','('])],[list(2,_,_),
     item(['corp','ireland'],'nnp')],[item(_,['vbd',')']),list(2,_,_),
     item(_,['nnp','vbn'])]).
rule(acquisition, purchaser, 5, 0,
     [],[item(['henley','allwaste','county','finlays'],'nnp'),item(_,'nnp')],
     [item(_,[')','vbd'])]).
rule(acquisition, purchaser, 4, 0,
     [item(_,[':','('])],[item(_,'nnp'),list(2,_,_),item(['go','capital'],_),
     item(['inc','corp'],'nnp')],[]).
rule(acquisition, purchaser, 5, 0,
     [item(_,['nns','cd','nn']),item(_,['to',':','nn'])],[list(3,_,'nnp'),
     item(['taft','inc','co'],'nnp')],[item(_,_),item(['columbia','it','narragansett'],_),
     item(_,[',','vbd','nnp']),item(_,'nnp')]).
rule(acquisition, purchaser, 4, 0,
     [item(_,[':','prp$','('])],[list(2,_,'nnp'),item(['bancshares','first',
     'holding'],'nnp'),item(['inc','ab'],'nnp')],[]).
rule(acquisition, purchaser, 4, 0,
     [item(['25','30','}'],_),item(['-','said'],_)],[list(2,_,'nnp'),
     item('corp','nnp')],[]).
rule(acquisition, purchaser, 3, 0,
     [],[item(['gst','amoskeag'],'nnp'),item(_,'nnp'),list(2,_,'nnp')],[item(_,'vbd')]).
rule(acquisition, purchaser, 7, 0,
     [item(_,['cd','pos','vbd']),item(['-','g.d.','its'],_)],[list(3,_,_),
     item(['ltd','co'],'nnp')],[item(['{','subsidiary','unit'],_)]).
rule(acquisition, purchaser, 5, 0,
     [item(_,[')',':',','])],[item(_,'nnp'),list(2,_,_),
     item(['technology','electronics','and','co'],_),
     item(['partners','co','communications','of'],_),item(_,'nnp')],[]).
rule(acquisition, purchaser, 4, 0,
     [item(_,['dt','nnp','nn']),list(2,['newly','units','3','shipping'],_),
     item(_,['vbn',':','to'])],[item(_,'nnp'),item(['central','industries','r.'],'nnp'),
     item(_,'nnp')],[]).
rule(acquisition, purchaser, 3, 0,
     [item(['to','march'],_),item(_,_),item(_,['pos',':'])],[list(3,_,'nnp'),
     item('plc','nnp')],[item('{','('),item(_,'nnp'),item('}',')'),
     item(_,['vbg','vbd']),list(2,_,_),item(_,['nn','to'])]).
rule(acquisition, purchaser, 3, 0,
     [item(_,[':','in'])],[item(_,['nnp','dt']),item(['-','leveraged'],_),item(_,_),
     item(['corp','firm'],_)],[item(_,'vbd'),item(_,['prp','vbn'])]).
rule(acquisition, purchaser, 7, 0,
     [item(['23','19','agreement'],_),item(['-','with'],_)],[list(2,_,'nnp'),
     item(['corp','inc'],'nnp')],[item(['said','{'],_),item(_,['prp','nnp']),
     item(['acquired','has','}'],_)]).
rule(acquisition, purchaser, 6, 0,
     [item(_,'nnp'),item(',',','),item(_,'nnp'),item(_,'cd'),item('-',':')],
     [list(2,_,'nnp'),item(['appleton','co'],'nnp')],[item(_,['vbd',',']),
     item(_,['prp','dt']),item(_,['vbd','nnp']),list(2,_,_)]).
rule(acquisition, purchaser, 4, 0,
     [item('march','nnp'),item(_,'cd'),item('-',':')],[list(2,_,'nnp'),
     item(['corp','inc'],'nnp')],[item('said','vbd'),item('it','prp'),
     item(['made','has'],_),item(['a','agreed'],_)]).
rule(acquisition, purchaser, 4, 0,
     [item(['march','bid','3'],_),item(_,_),item(_,['(',':'])],[list(2,_,['pos','nnp']),
     item(['grandview','holdings','hospitality'],'nnp'),item(['plc','inc'],'nnp')],
     []).
rule(acquisition, purchaser, 3, 0,
     [item(['13','said'],_),item(['-','that'],_)],[list(3,_,'nnp')],[item(_,'vbd'),
     list(2,_,_),item(_,['dt','jj'])]).
rule(acquisition, purchaser, 3, 0,
     [item(_,['to','nns']),item(_,_),item(['with','by'],'in')],[list(2,_,'nnp'),
     item(['inc','corp'],'nnp')],[item(_,[',','vbd']),list(2,_,_),
     list(2,['-','its','florida'],_),item(_,['nn','vbn'])]).
rule(acquisition, purchaser, 3, 0,
     [item(_,['cc','to']),item(['s.','{'],_)],[list(2,_,['nnp','jjs']),
     item(['rockwell','inc'],'nnp')],[]).
rule(acquisition, purchaser, 3, 0,
     [item(_,['nn','cd']),item(_,['(',':'])],[item(_,_),item(['holdings','travellers'],_),
     item(['ltd','corp'],'nnp')],[]).
rule(acquisition, purchaser, 4, 0,
     [item(['-','24'],_),item(_,['(',':'])],[list(2,_,'nnp'),item(['corp','ltd'],'nnp')],
     [item(_,[')','vbd']),item(_,['vbd','prp']),item(['it','has'],_),
     item(['is','agreed'],_)]).
rule(acquisition, purchaser, 3, 0,
     [item(['known','acquired'],'vbn'),item(['as','by'],'in')],[list(3,_,'nnp'),
     item(['international','inc'],'nnp')],[]).
rule(acquisition, purchaser, 4, 0,
     [item(_,['in','nn',':']),item(_,['nn',':','('])],[list(2,_,'nnp'),
     item(['j.','american','pty'],'nnp'),item(_,'nnp')],[]).
rule(acquisition, purchaser, 5, 0,
     [item(_,['in','vbg','cd']),list(2,_,_),item(_,['nn','vbn','jj']),
     item(['that','to','insurer'],_)],[list(3,_,'nnp'),item(_,_),
     item(['inc','corp','n.v.'],'nnp')],[item(_,['in','('])]).
rule(acquisition, purchaser, 4, 0,
     [item(['of','-'],_)],[item(_,'nnp'),item(_,_),item(_,'nnp'),item(_,_),
     item(['kellogg','inc'],'nnp')],[item(_,'vbd')]).
rule(acquisition, purchaser, 3, 0,
     [item(['stressed','27'],_),item(['that','-'],_)],[list(4,_,_)],
     [item(_,['cc','vbd'])]).
rule(acquisition, purchaser, 5, 0,
     [],[item(['bryson','siebel','prospect','kappa'],'nnp'),list(3,_,_),
     item(['inc','plc'],_)],[]).
rule(acquisition, purchaser, 7, 0,
     [item(['and','-','by','{'],_)],[item(['dudley','first','oregon','union'],'nnp'),
     list(2,_,'nnp'),item(['taft','corp','mills','sa'],'nnp')],[list(3,_,_),
     item(_,['vbn','jj','vbd'])]).
rule(acquisition, purchaser, 4, 0,
     [item(['19','5'],_),item('-',':')],[item(_,'nnp'),list(2,_,['nnp',':']),
     item(['co','lp'],'nnp')],[item(_,['vbd','nnp'])]).
rule(acquisition, purchaser, 4, 0,
     [item(_,'cd'),item('-',':')],[item(_,'nnp'),item(_,'nnp'),item(_,'nnp')],
     [item(['said',','],_),item(_,['in','dt']),item(_,['dt','nnp']),
     list(2,_,'nn'),item(_,['in',','])]).
rule(acquisition, purchaser, 4, 0,
     [item(['.','24','corp'],_),item(_,['endsent',':','pos'])],[item(_,'nnp'),
     item(_,'nnp'),item(_,'nnp')],[item(['said','unit'],_),
     item(['in','it','said'],_),item(_,['dt','vbz','prp'])]).
rule(acquisition, purchaser, 4, 0,
     [item('investor','nn')],[item(_,'nnp'),item(_,'nnp')],
     [item(_,['vbd',','])]).
rule(acquisition, purchaser, 3, 0,
     [item(['-','with'],_),item('{','(')],[list(2,_,'nnp'),
     item(['santander','plc'],_)],[]).
rule(acquisition, purchaser, 3, 0,
     [item(_,['dt','nnp']),item(_,_),item(_,['(','endsent'])],[list(3,_,'nnp'),
     item(['negara','inc'],'nnp')],[item(_,[')','vbd']),item(_,['vbd','prp'])]).
rule(acquisition, purchaser, 3, 0,
     [item(_,['cd','nn']),item(_,[':','to'])],[item(_,'nnp'),item(_,'nnp'),
     item(_,'nnp'),item('inc','nnp')],[item(_,['vbd','(']),list(2,_,_),
     item(_,['rb',')'])]).
rule(acquisition, purchaser, 4, 0,
     [item(['the','by','{'],_)],[item(_,'nnp'),item(['retrieval','industries',
     'alden'],'nnp'),list(2,_,'nnp')],[item(_,['.','in',')'])]).
rule(acquisition, purchaser, 7, 0,
     [item(_,['nn',':','('])],[item(_,'nnp'),list(2,['department','t.','stores',
     'coal'],'nnp'),item(['sosnoff','co'],'nnp')],[item(_,['.','vbd',')'])]).
rule(acquisition, purchaser, 5, 0,
     [list(2,_,_),list(2,_,_)],[item(_,'nnp'),list(2,_,'nnp'),
     item(['management','partners'],'nnp'),item(_,'nnp')],[]).
rule(acquisition, purchaser, 3, 0,
     [item(_,['to',':']),item('{','(')],[list(3,_,'nnp'),item(['ag','plc'],'nnp')],
     []).
rule(acquisition, purchaser, 3, 0,
     [item(['-','company'],_)],[item(_,'nnp'),item(['pacific','finanziara'],'nnp'),
     item(_,'nnp')],[]).
rule(acquisition, purchaser, 3, 0,
     [],[item(['215','narragansett'],_),item(_,'nnp'),item(_,'nnp')],[]).
rule(acquisition, purchaser, 3, 0,
     [item(['affiliate','9'],_),item(_,[',',':'])],[item(_,'nnp'),
     item(['inc','co'],'nnp')],[]).
rule(acquisition, purchaser, 3, 0,
     [item(['april','announced'],_),item(_,_),item(_,[':','('])],[list(2,_,'nnp'),
     item(['sa','ltd'],'nnp')],[list(2,_,_),item(_,['dt','('])]).
rule(acquisition, purchaser, 4, 0,
     [item(_,[':','(','to'])],[list(3,_,['nnps','nnp']),
     item(['canada','products','usa'],'nnp'),item(_,'nnp')],
     [item(_,[',',')','in'])]).
rule(acquisition, purchaser, 3, 0,
     [item(['president','investor'],'nnp')],[item(_,'nnp'),item(_,'nnp')],
     [item(_,['vbz','vbd']),item(['been','he'],_)]).
rule(acquisition, purchaser, 4, 0,
     [],[item(['lone','nestle','malaysia'],'nnp'),list(2,_,_),
     item(['inc','ltd','bank'],_)],[]).
rule(acquisition, purchaser, 3, 0,
     [item(['developer','a'],_)],[item(_,'nnp'),item(['trump','corp'],'nnp')],[]).
rule(acquisition, purchaser, 3, 0,
     [item(['2','13'],'cd'),item('-',':')],[item(_,'nnp'),list(3,_,_),
     item(['co','barbara'],'nnp')],[item('said','vbd')]).
rule(acquisition, purchaser, 3, 0,
     [],[item(['metropolitan','groupe'],'nnp'),item(_,'nnp'),item(_,'nnp'),
     item(_,'nnp')],[]).
rule(acquisition, purchaser, 4, 0,
     [],[item(['upland','ge','mcandrews'],'nnp'),list(3,_,_),
     item(['corp','inc'],'nnp')],[]).
rule(acquisition, purchaser, 3, 0,
     [item(['-','26'],_),item(_,['(',':'])],[list(2,_,'nnp'),item('inc','nnp')],
     [item(_,['sym','vbd'])]).
rule(acquisition, purchaser, 3, 0,
     [item(_,[':','vbn'])],[item(_,'nnp'),item(_,'nnp'),list(2,_,_),
     item(['bank','investment'],'nnp'),item('corp','nnp')],[]).
rule(acquisition, purchaser, 3, 0,
     [],[item(['snyder','faraway'],'nnp'),item(_,'nnp'),item(_,'nnp'),item(_,'nnp')],
     []).
rule(acquisition, purchaser, 4, 0,
     [item(_,[':',','])],[list(3,_,'nnp'),item(['bancorp','n.v.'],'nnp')],[]).
rule(acquisition, purchaser, 3, 0,
     [item(['into','to'],_)],[item(_,'nnp'),item(_,'nnp'),item(['plc','inc'],'nnp')],
     []).
rule(acquisition, purchaser, 3, 0,
     [item(['11','times'],_),item(['-','that'],_)],[item(_,'nnp'),list(2,_,_),
     item(['co','inc'],'nnp')],[item(['said','has'],_),list(2,_,_),item(_,['vbz','in'])]).
rule(acquisition, purchaser, 4, 0,
     [item(_,[':','in'])],[list(2,_,'nnp'),item(['acquisition','group'],'nnp'),
     item(['co','plc'],'nnp')],[]).
rule(acquisition, purchaser, 3, 0,
     [item(_,['in','pos'])],[list(2,_,'nnp'),item(['industries','and'],_),
     item(['inc','foods'],_)],[item(['{','subsidiary'],_)]).
rule(acquisition, purchaser, 4, 0,
     [item(['30','18'],'cd'),item('-',':')],[item(_,'nnp'),
     item(['financial','group'],'nnp'),item(_,'nnp')],[]).
rule(acquisition, purchaser, 3, 0,
     [item(_,['nn','nnp']),item(_,_),item(_,_)],[item(_,'nnp'),item(['and','coal'],_),
     item(_,_),list(2,_,_),item('ltd','nnp')],[]).rule(acquisition, purchaser, 3, 0,
     [],[item(['pantera','gli'],'nnp'),item(_,['pos','nnp']),item(_,'nnp')],[]).
rule(acquisition, purchaser, 3, 0,
     [],[item(_,'nnp'),item(['kgaa','health'],'nnp'),
     list(2,['inc','of','affiliates'],_)],[item(_,_),item(_,_),item(_,['nnp','prp'])]).
rule(acquisition, purchaser, 3, 0,
     [item(_,['vbn',':'])],[item(_,'nnp'),item(['industries','holdings'],'nnp'),
     item(_,'nnp')],[item(_,['nnp','('])]).
rule(acquisition, purchaser, 4, 0,
     [item(['29','agreement'],_),item(['-','with'],_)],[list(2,_,'nnp'),
     item(['inc','co'],'nnp')],[item(_,['vbd','in'])]).
rule(acquisition, purchaser, 3, 0,
     [item(_,['(',':'])],[item(['dart','vertex'],'nnp'),item(_,'nnp'),item(_,'nnp')],
     []).
rule(acquisition, purchaser, 3, 0,
     [item(_,['cd','nnp']),item(_,_)],[list(2,_,'nnp'),item(['''s','partners'],_),
     item(['inc','lp'],'nnp')],[]).
rule(acquisition, purchaser, 5, 0,
     [item(['.','18'],_),item(_,['endsent',':'])],[item(_,'nnp'),
     list(2,_,_),item(['corp','info'],'nnp')],[item([',','said'],_)]).
rule(acquisition, purchabr, 6, 0,
     [item(_,['nnp','rb','nn']),item(_,['vbn','nn']),item(_,[':','.']),
     item(_,['(','endsent'])],[list(2,_,'nnp')],[item(['}','expects','said'],_),
     item(['signed','last','it'],_)]).
rule(acquisition, purchabr, 3, 0,
     [],[item(['macandrews','community'],'nnp'),item(_,_),item(['forbes','system'],_)],
     []).
rule(acquisition, purchabr, 4, 0,
     [],[item(['chase','esselte','gabelli'],'nnp')],[item(_,['nnp','nn'])]).
rule(acquisition, purchabr, 4, 0,
     [item(['york','dlrs','endsent'],_),list(2,_,_),list(2,_,_),
     item(['27','.','circumstances'],_),item(_,[':','endsent',','])],
     [list(2,_,['rb','nnp'])],[item(['international',',','said'],_),
     item(['holdings','a','he'],_)]).
rule(acquisition, purchabr, 4, 0,
     [],[item(_,'nnp'),item(['food','-'],_),list(2,_,'nnp')],[item(['{','co'],_),
     item(_,'nnp'),item(_,_),item(_,['vbz','prp']),item(_,_),item(_,_),item(_,'nnp')]).
rule(acquisition, purchabr, 12, 0,
     [],[item(['packaging','sanwa','thermo','united','coca','rio','pantera','dudley',
     'general'],'nnp'),item(['systems','bank','process','insurance','cola','tinto',
     '''s','taft','acquisition'],_)],[]).
rule(acquisition, purchabr, 4, 0,
     [],[item(['swiss','lvi','safeguard'],'nnp'),item(_,'nnp')],[item(_,_),
     item(_,['nnp','vb'])]).
rule(acquisition, purchabr, 3, 0,
     [item(_,['dt','endsent'])],[item(_,['nnp','nn']),list(3,_,['prp','nn',':']),
     item(['taft','l'],'nnp')],[]).
rule(acquisition, purchabr, 3, 0,
     [],[item(_,'nnp'),item(_,'nnp')],[item(_,['vbz','(']),item(['stake','ht'],_),
     item(_,_),item(_,['nnp','vbz'])]).
rule(acquisition, purchabr, 4, 0,
     [],[item(['bryson','dart','teck'],'nnp')],[item(['said','corp'],_)]).
rule(acquisition, purchabr, 6, 0,
     [item(_,['jj','in']),list(2,_,_),item(_,_),item(_,_),item(_,['in','nnp']),
     item(_,_),item(_,[',','prp$'])],[item(_,'nnp')],[item(['said','furniture'],_),
     item(_,['.','nnp']),list(2,_,_),item(_,['nnp','vbd'])]).
rule(acquisition, purchabr, 6, 0,
     [list(2,_,_)],[item(['radiation','dixons','hearst','lsb'],'nnp')],[]).
rule(acquisition, purchabr, 4, 0,
     [item(_,[':','cd']),item(_,['(',':'])],[item(_,'nnp')],
     [item(['america','ltd'],'nnp')]).
rule(acquisition, purchabr, 3, 0,
     [],[item(_,'nnp'),item(_,'nnp')],[item('co','nnp'),item(_,['(','vbd']),
     item(_,'nnp'),item(_,_),item(_,['to','nnp'])]).
rule(acquisition, purchabr, 5, 0,
     [item(_,['in','nnp']),item(_,_),item(['york','march'],'nnp'),item(_,_),
     item(['saul','-'],_)],[item(_,'nnp')],[item(['told','group'],_),
     item(_,_),list(2,_,_),item(_,_),item(_,['nnp','cd'])]).
rule(acquisition, purchabr, 3, 0,
     [item(['china','agreement'],_),item(_,['vbz',','])],[list(2,_,['nnps','nnp'])],
     [item(['buys','will'],_)]).
rule(acquisition, purchabr, 4, 0,
     [],[item(['metropolitan','henley','cross'],'nnp')],[item(_,['(','nn','vbd'])]).
rule(acquisition, purchabr, 3, 0,
     [item(_,['(',':'])],[item(_,'nnp'),item(['montauk','''s'],_),
     item(_,['nns','nnp'])],[item(_,[')','nnp'])]).
rule(acquisition, purchabr, 3, 0,
     [item(['13','-'],_),item(['-','{'],_)],[list(2,['''s','cellular','scott'],_),
     item(_,'nnp')],[item('inc','nnp')]).
rule(acquisition, purchabr, 5, 0,
     [item(_,['nn','in','dt'])],[item(['twa','sosnoff','searle','coke'],_)],
     [item(['action','''s','held','.'],_)]).
rule(acquisition, purchabr, 5, 0,
     [],[item(['ufurf','snyder','conagra','c.o.m.b.'],_)],[]).
rule(acquisition, purchabr, 4, 0,
     [item(_,['cd','nn']),item(['-','of'],_)],[list(2,_,'nnp')],
     [item(['corp','''s'],_),item(_,_),item(['.','darta.o','in'],_)]).
rule(acquisition, purchabr, 4, 0,
     [item(['2','january','-'],_),item(_,[':',',','('])],[item(_,'nnp')],
     [item(['said','reported','oil'],_)]).
rule(acquisition, purchabr, 4, 0,
     [item(['june','exchange'],'nnp'),item(['1','commission'],_),item(_,[':',','])],
     [list(3,_,[':','nnp'])],[item(['corp',','],_)]).
rule(acquisition, purchabr, 4, 0,
     [item(['ohio','n.h.',','],_),item(_,_),item(['june','march',','],_),
     item(_,['cd','cc']),item(_,[':','prp$'])],[list(2,_,'nnp'),item(_,'nnp')],
     [item(['corp','bank'],'nnp')]).
rule(acquisition, purchabr, 8, 0,
     [],[item(['neoax','clabir','3m','contel','narragansett','amsouth'],_)],[]).
rule(acquisition, purchabr, 5, 0,
     [item(_,[',','dt','endsent','in'])],
     [list(2,['to','roth','j.p.','-','singapore','stars'],_),
     item(['american','industries','government','go'],_)],[item(['has','said','inc'],_)]).
rule(acquisition, purchabr, 3, 0,
     [item(['chicago','march'],'nnp'),item(['corp','18'],_),item(_,['pos',':'])],
     [item(_,'nnp'),item(_,'nnp')],[item(['bank','corp'],'nnp'),
     item(_,['nn','vbd'])]).
rule(acquisition, purchabr, 4, 0,
     [item(_,['to','jj']),item(_,['vb','.']),item(_,['in','endsent'])],
     [item(_,'nnp')],[item(_,['in','cc'])]).
rule(acquisition, purchabr, 3, 0,
    [item(_,[’.’,’:’]),item([’endsent’,’two’],_)],[list(2,_,’nnp’)],
    [item([’’’s’,’stock’],_),item(_,’nn’),item(_,[’nn’,’,’])]).
rule(acquisition, purchabr, 7, 0,
    [item([’agreement’,’19’,’investor’],_),item([’for’,’-’,’mulls’],_)],[item(_,’nnp’)],
    [item(_,_),list(2,[’acquire’,’lp’,’kpa’,’energies’],_),item(_,[’dt’,’)’,’vbd’]),
    item(_,[’cd’,’prp’,’:’])]).
rule(acquisition, purchabr, 4, 0,
    [item([’’’’,’25’,’bnp’],_),item(_,[’prp’,’:’,’in’])],[list(2,_,[’nn’,’pos’,’nnp’])],
    [item([’enterprises’,’corp’,’bid’],_)]).
rule(acquisition, purchabr, 7, 0,
    [item(_,[’nn’,’in’,’rb’]),item(_,_),item([’"’,’two’,’.’],_),
    item(_,[’dt’,’jj’,’endsent’])],[list(3,_,[’cc’,’nnp’])],
    [item([’group’,’shares’,’said’],_),item([’told’,’.’,’it’],_),
    item(_,[’dt’,’endsent’,’vbd’]),list(2,_,_),item(_,[’.’,’dt’,’in’])]).
rule(acquisition, purchabr, 3, 0,
    [item([’louis’,’april’],’nnp’),item([’-’,’17’],_),item([’based’,’-’],_)],
    [item(_,’nnp’),item(_,[’nnps’,’nnp’])],[]).
rule(acquisition, purchabr, 3, 0,
    [],[item(_,’nnp’),item(_,_),item([’forbes’,’barbara’],’nnp’)],[]).
rule(acquisition, purchabr, 3, 0,
    [item([’of’,’26’],_),item(_,[’dt’,’:’])],[item(_,’nnp’)],
    [item([’group’,’international’],’nnp’),item(_,[’in’,’nnp’])]).
rule(acquisition, purchabr, 3, 0,
    [],[item([’trus’,’metropolitan’],’nnp’),list(2,_,’nnp’)],[item([’inc’,’{’],_),
    item(_,[’(’,’nnp’])]).
rule(acquisition, purchabr, 3, 0,
    [item([’30’,’to’],_),item(_,[’:’,’dt’])],[item(_,’nnp’)],[item([’bank’,’facility’],_)]).
rule(acquisition, purchabr, 4, 0,
    [item(_,[’endsent’,’vbd’]),list(2,_,_),list(2,_,_),list(2,_,_),item([’-’,’by’],_)],
    [item(_,’nnp’)],[item([’mining’,’.’],_),item([’co’,’endsent’],_)]).
rule(acquisition, purchabr, 5, 0,
    [item([’issued’,’5’],_),item([’by’,’-’],_)],[list(3,_,[’:’,’nnp’])],
    [item([’.’,’corp’],_)]).
rule(acquisition, purchabr, 4, 0,
    [item([’is’,’amount’,’april’],_),item([’subject’,’.’,’1’],_),item(_,_)],
    [list(2,_,[’nnps’,’nnp’])],[item([’’’s’,’said’,’and’],_),
    item([’ability’,’the’,’chemicals’],_)]).
rule(acquisition, purchabr, 5, 0,
    [item(_,[’nnp’,’:’]),item([’w’,’said’,’{’],_)],[item(_,’nnp’)],
    [item([’inc’,’is’,’ltd’],_)]).
rule(acquisition, purchabr, 3, 0,
    [item(_,’nn’),item(_,[’to’,’,’])],[item(_,’nnp’)],[item([’corp’,’said’],_),
    item([’for’,’it’],_)]).
rule(acquisition, purchabr, 3, 0,
    [item(_,[’.’,’cd’]),item(’-’,’:’),item(_,[’vbn’,’(’])],[list(2,_,’nnp’)],
    [item([’said’,’group’],[’vbd’,’nnp’])]).
rule(acquisition, purchabr, 3, 0,
    [item(_,[’in’,’nn’]),item(_,[’nn’,’in’]),item([’of’,’some’],_)],[item(_,’nnp’)],
    [item([’’’s’,’coffee’],_),list(2,_,_),item(_,_),item(_,[’vbp’,’nnp’])]).
rule(acquisition, purchabr, 3, 0,
    [item(_,[’nn’,’endsent’]),item([’.’,’dna’],_),item([’endsent’,’said’],_)],
    [list(2,_,’nnp’)],[item(_,[’pos’,’md’]),item(_,[’cd’,’vb’])]).
rule(acquisition, purchabr, 5, 0,
    [],[item([’alza’,’usair’,’deak’],’nnp’)],[item([’corp’,’’’s’,’said’],_),
    list(2,_,_),item(_,[’nnp’,’vb’,’endsent’])]).
rule(acquisition, purchabr, 3, 0,
    [item([’but’,’{’],_)],[item(_,’nnp’)],[item([’said’,’industries’],_),
    item(_,_),item(_,_),item(_,[’nnp’,’vbd’])]).
rule(acquisition, purchabr, 3, 0,
    [item(_,[’in’,’:’])],[list(2,_,’nnp’),item([’mortgage’,’capital’],’nnp’)],
    [list(2,_,_),item(_,_),item(_,[’jj’,’vbd’])]).
rule(acquisition, purchabr, 3, 0,
    [item(_,['in','('])],[list(2,_,'nnp'),item(['microwave','capital'],'nnp')],
    [item(_,_),item([',','l.p.'],_)]).
rule(acquisition, purchabr, 5, 0,
    [item(['june','shares','.'],_),item(_,_),item(_,['endsent',':'])],[item(_,'nnp')],
    [item(['inc','said','also'],_),item(_,_),item(_,['nns','prp'])]).
rule(acquisition, purchabr, 3, 0,
    [item(_,['nnp','cd']),item(_,[')',':']),item(['as','{'],_)],[item(_,'nnp')],
    [item(_,'nnp'),item(['ltd','}'],_),item(_,['(','vbd'])]).
rule(acquisition, purchabr, 4, 0,
    [item(_,['jj','nn']),list(2,_,_),item(_,_),item(['by','endsent'],_)],[item(_,'nnp')],
    [item(['.','said'],_),item(_,['endsent','dt']),item(['the','acquisition'],_)]).
rule(acquisition, purchabr, 6, 0,
    [item(_,['vbn','nn']),item(['by','-'],_)],[list(2,_,'nnp')],[item(['.','corp'],_),
    item(_,['endsent','vbd']),item(_,_),item(_,_),item(_,_)]).
rule(acquisition, purchabr, 3, 0,
    [item(_,['cd','nns']),item(_,['nn','.']),item(['of','endsent'],_)],[item(_,'nnp')],
    [item(['for','said'],_),item(_,['dt','nn']),item(['hughes','of'],_)]).
rule(acquisition, purchabr, 4, 0,
    [],[item(['monier','allwaste','grandview'],'nnp')],[]).
rule(acquisition, purchabr, 4, 0,
    [],[item(['furniture','chrysler','emery'],'nnp')],[]).
rule(acquisition, purchabr, 3, 0,
    [item(_,'nnp'),item(['to','with'],_)],[item(_,'nnp')],[item(_,['(','endsent'])]).
rule(acquisition, purchabr, 4, 0,
    [item(['commission','19'],_),item(_,[',',':'])],[item(_,'nnp')],
    [item(['said','corp'],_),item(_,['prp$','vbd']),item(_,['nn','prp'])]).
rule(acquisition, purchabr, 4, 0,
    [],[item(['merrill','unicorp','globe'],'nnp'),list(2,_,'nnp')],
    [item(['{','was','manufacturing'],_)]).
rule(acquisition, purchabr, 6, 0,
    [item(_,['nnp','vbn']),item(_,_),item(_,['nnp','prp']),item(_,['cd','.']),
    item(_,[':','endsent'])],[item(_,'nnp')],[item(['co','''s'],_),
    list(2,_,_),item(_,['prp','vbd'])]).
rule(acquisition, purchabr, 3, 0,
    [item(_,['dt','vbd']),item(['acquisition','that'],_)],[item(_,'nnp')],
    [item(_,['pos','cc'])]).
rule(acquisition, purchabr, 3, 0,
    [item(_,['(','nn']),list(2,_,_),item(['unit','-'],_),item(_,_)],[item(_,'nnp')],
    [item(['holdings','benefitting'],_),item(_,['nnp','in'])]).
rule(acquisition, purchabr, 3, 0,
    [item(['electric','exchange'],'nnp'),item(['co','commission'],'nnp'),
    item(_,['pos',','])],[list(2,_,'nnp')],[item(['corp','said'],_),
    item(_,['vbd','prp']),list(2,_,_),item(_,_),item(_,['to','dt'])]).
rule(acquisition, purchabr, 3, 0,
    [item(['30','2'],'cd'),item('-',':')],[item(_,'nnp')],[item(_,'nnp'),
    item(['corp',','],_)]).
rule(acquisition, purchabr, 3, 0,
    [item(['-','on'],_)],[list(2,_,'nnp')],[item(['bancshares','offer'],_)]).
rule(acquisition, purchabr, 3, 0,
    [item(_,[':','endsent'])],[item(_,'nnp')],[item(['capital','publishes'],_)]).
rule(acquisition, purchabr, 3, 0,
    [item(['endsent','june'],_),item(_,['nnp','cd']),item(_,['vbd',':'])],
    [item(_,'nnp')],[item(['''s','department'],_)]).
rule(acquisition, purchabr, 5, 0,
    [item('endsent','endsent')],[item(_,'nnp')],[item(['already','also'],'rb'),
    item(['holds','reported'],_)]).
rule(acquisition, purchabr, 3, 0,
    [item(['-','than'],_)],[item(_,'nnp')],[item(['financial','that'],_)]).
rule(acquisition, purchabr, 3, 0,
    [],[item(['american','first'],_),item(_,['nns','nnp'])],[item(_,['vbg','.'])]).
rule(acquisition, purchabr, 3, 0,
    [],[item(['union','jannock'],'nnp'),item(_,'nnp')],[item(_,'nnp')]).
```

```
rule(acquisition, purchabr, 3, 0,
     [item(_,['in','vbd']),item(_,_),item(_,['cd','endsent'])],[list(2,_,'nnp')],
     [item(['said','staff'],_),item(['it','will'],_)]).
rule(acquisition, purchabr, 3, 0,
     [item(_,['rb','nnp']),item(_,['in','nnp']),item(_,_),item(_,_),item(_,['nns','cd']),
     item(_,[',',':'])],[item(_,'nnp')],[item(['''s','ltd'],_),item(_,_),list(2,_,_),
     item(['taft','said'],_)]).
rule(acquisition, purchabr, 3, 0,
     [list(2,_,_)],[item(['fleet','norstar'],'nnp')],[item(_,['nn','nnp']),
     item(_,['nnp','vbd'])]).
rule(acquisition, purchabr, 3, 0,
     [item(_,['nnp','nn']),list(2,[',','has','association'],_),item(['succeeded','by'],_)],
     [list(2,_,'nnp')],[item(_,['in',','])]).
rule(acquisition, dlramt, 5, 0,
     [item(_,['nn',')']),list(2,_,_),item(['estimated','for'],_)],[item(_,'cd'),
     item('mln','jj'),item(_,['nn','nns'])],[item(_,['nn','.']),
     item(_,_),item(_,['endsent','dt'])]).
rule(acquisition, dlramt, 8, 0,
     [item(_,['vb','nnp']),item(['for','bids'],_)],[list(2,_,['jj','cd']),
     item(_,['nnp','jj']),item(['stg','dlrs'],_)],[item(_,_),item(_,_),item(_,_),
     item(_,['nn','nnp'])]).
rule(acquisition, dlramt, 7, 0,
     [item(['9','valued','stock'],_),item(['at','for'],'in')],[list(2,_,[',','cd']),
     item(['mln','691'],_),item('dlrs','nns')],[]).
rule(acquisition, dlramt, 10, 0,
     [item(_,['nn','nnp']),item(_,_),item(_,_),item(_,['nn','nnp']),list(2,_,_),
     item(_,['dt','in'])],[item('undisclosed','jj')],[item(['sum','terms'],_)]).
rule(acquisition, dlramt, 6, 0,
     [item(_,'nnp'),item(['rejects','for'],_)],[item(_,'cd'),item('mln',_),
     item(['stg','dlrs'],_)],[item(_,['nn',','])]).
rule(acquisition, dlramt, 8, 0,
     [item(_,_),item(_,['nnp','endsent']),item(_,_),item(_,_)],[item('undisclosed','jj')],
     [item(['terms','.'],_),item(_,_),item(_,['endsent','prp'])]).
rule(acquisition, dlramt, 9, 0,
     [item(_,'dt'),item(_,_),item(['of','were'],_)],[item(['not','10.5'],_),
     list(2,['dlrs','disclosed','mln'],_)],[item('.','.')]).
rule(acquisition, dlramt, 3, 0,
     [],[item(['94.8','82'],'cd'),item('mln','jj'),item(_,_)],[]).
rule(acquisition, dlramt, 8, 0,
     [item(_,['nnp',')']),item(_,[',','in']),item(['for','an'],_)],
     [item('undisclosed','jj')],[]).
rule(acquisition, dlramt, 3, 0,
     [],[item(['1.62','538'],'cd'),item(_,_),item('dlr','nn')],[]).
rule(acquisition, dlramt, 3, 0,
     [item(_,'in'),item(_,['in','vbg'])],[item(['one','100'],'cd'),
     item(_,_),item(['stg','dlrs'],_)],[item(['.','in'],_)]).
rule(acquisition, dlramt, 5, 0,
     [],[item(_,'cd'),item(['billion','mln'],'nnp'),item('dlrs','nns')],[]).
rule(acquisition, dlramt, 8, 0,
     [item(_,['nn','nnp']),item(['were','an'],_)],[item(['not','undisclosed'],_),
     list(2,['price','disclosed','purchase'],_)],[item(_,['.','in'])]).
rule(acquisition, acquired, 3, 0,
     [item(['16','schedule'],_),item(_,[':','in'])],[list(3,_,'nnp')],
     [item(['said','merger'],_)]).
rule(acquisition, acquired, 3, 0,
     [item(_,['to','vb']),item(['purchase','its'],_)],[item(_,'nnp'),item(_,'nnp')],
     [item(_,[',','nn'])]).
rule(acquisition, acquired, 4, 0,
     [item(['of','its'],_)],[list(2,_,'nnp'),item(['airlines','inc','operations'],_)],
     [item(['for','to'],_)]).
rule(acquisition, acquired, 5, 0,
     [item(_,['in','('])],[list(2,['plastics','cenergy','western'],_),
     item('corp','nnp')],[]).
rule(acquisition, acquired, 4, 0,
     [item(['the','its'],_)],[list(2,_,_),item(['magnetics','division'],_)],
     [item(_,['nn','to'])]).
rule(acquisition, acquired, 5, 0,
     [],[item(['integrated','foote','sunrise','com'],'nnp'),list(3,_,_),
     item(['branch','co','inc'],_)],[]).
rule(acquisition, acquired, 5, 0,
     [item(['in','its'],_)],[item(_,'nnp'),item(['enterprises','corp'],_)],[]).
rule(acquisition, acquired, 4, 0,
     [],[item(['consolidated','group','micro'],_),list(2,_,_),
     item(['international','mineral','ii'],_),item(_,_)],[]).
rule(acquisition, acquired, 3, 0,
     [item(['may','acquisition'],_),item(['support','of'],_)],[list(2,_,['nn','nnp','cd']),
     item(['corp','centers'],_)],[]).
rule(acquisition, acquired, 4, 0,
     [item(_,['jj',',','dt']),item(_,_),item(['in','buy','of'],_)],
     [item(['johnson','first','ranks'],'nnp'),item(_,'nnp'),list(2,_,_)],
     [item(_,['(','.','in'])]).
rule(acquisition, acquired, 8, 0,
     [item(['in','acquired','purchase'],_)],[list(4,_,['vbg','nn','dt','nnp']),
     item(['financial','plant','corp'],_)],[item(_,['to','in',','])]).
rule(acquisition, acquired, 3, 0,
     [item(_,['in','nnp']),list(2,['-','{','4'],_)],[item(['usair','pioneer'],'nnp'),
     list(2,_,'nnp'),item(['inc','ltd'],'nnp')],[]).
rule(acquisition, acquired, 4, 0,
     [item(['in','bid'],_),item(_,['(','in'])],[list(3,_,'nnp'),
     item(['corp','inc'],'nnp')],[item(_,[')',','])]).
rule(acquisition, acquired, 5, 0,
    [],[item(['salt','south','coffee','d'],_),list(3,_,_),
     item(['division','advocate','plant','corp'],_)],[]).
rule(acquisition, acquired, 3, 0,
     [item(_,['prp$','in'])],[list(3,_,'nnp'),item(['creek','division'],'nnp')],
     [item(_,['to','nn'])]).
rule(acquisition, acquired, 8, 0,
     [],[item(['cyclops','hillards','datron'],'nnp'),item(_,'nnp')],[]).
rule(acquisition, acquired, 4, 0,
     [item(_,['nns','nnp','prp']),item(['of','and','told'],_)],[list(2,_,'nnp'),
     item(['corp','africa'],'nnp')],[item([',','said','{'],_),
     list(2,_,_),item(_,['nn','vbd',')'])]).
rule(acquisition, acquired, 3, 0,
     [item(['of','{'],_)],[item(['cenergy','protein'],'nnp'),list(2,_,'nnp')],
     [item(_,['.',')'])]).
rule(acquisition, acquired, 4, 0,
     [item(_,['in','nnp']),item(_,['dt',')'])],[list(2,_,'nnp'),item(_,'nnp'),
     item('division','nnp')],[]).
rule(acquisition, acquired, 5, 0,
     [item(['buy','with','in'],_)],[item(_,'nnp'),list(2,_,_),
     item(['restaurants','sa','inc.'],'nnp')],[]).
rule(acquisition, acquired, 3, 0,
     [item(_,['.','in']),item(_,['endsent','('])],[list(3,_,'nnp'),
     item(['corp','services'],_)],[item(_,['vbz',')']),item(_,['nn',','])]).
rule(acquisition, acquired, 3, 0,
     [item(_,['(','prp'])],[item(_,'nnp'),item(_,_),item(['bank','melrose'],'nnp'),
     list(2,['dickson','bhd',','],_)],[item(_,[')','cc'])]).
rule(acquisition, acquired, 3, 0,
     [item('of','in')],[list(2,_,'nnp'),item(['acquiring','services'],_),
     item(_,'nnp')],[]).
rule(acquisition, acquired, 5, 0,
     [],[item(_,'nnp'),item(['fairview','cable','lewis','baltic'],'nnp'),
     list(2,_,['nnps','nn','nnp'])],[item(_,[')','in'])]).
rule(acquisition, acquired, 15, 0,
     [item(_,['vb','in','nn']),item(['the','its','in'],_)],[item(_,['jj','nnp']),
     list(2,_,_),item(['project','portfolio','inc'],_)],[]).
rule(acquisition, acquired, 3, 0,
     [item(_,['nn','in']),item(['of','its'],_)],[list(2,_,_),item(['fund','division'],_)],
     [item(_,[',','to']),item(_,['"','dt'])]).
rule(acquisition, acquired, 4, 0,
     [item(['stake','merge'],_),item(_,'in')],[list(2,_,'nnp'),item('corp','nnp')],
     [item(['to','was'],_)]).
rule(acquisition, acquired, 3, 0,
     [item(_,['nn','vbz']),item(_,_),item(_,[',','dt'])],[list(2,_,'nnp'),
     item(['chronicle','co'],'nnp')],[item([',','from'],_)]).
rule(acquisition, acquired, 3, 0,
     [item(_,['(','in'])],[item(_,'nnp'),list(2,_,_),
     item(['services','investmento'],'nnp'),item(['inc','sarl'],'nnp')],[]).
rule(acquisition, acquired, 5, 0,
     [item(_,['cc','vb'])],[item(_,['nnp','cd']),item(_,_),
     item(['inc','hospitals'],_)],[item(_,['in','to'])]).
rule(acquisition, acquired, 4, 0,
     [item(['stake','of','affiliate'],_),item(_,['(','in'])],[item(_,'nnp'),
     list(2,_,_),item(_,['nnp','vbg']),item(['ltd','n.v.','ag'],'nnp')],
     [item(_,_),list(2,_,_),item(_,_),item(_,[',',':'])]).
rule(acquisition, acquired, 3, 0,
     [item(_,'nn'),item(_,['vbn','in'])],[list(2,_,'nnp'),
     item(['bancorp','technology'],'nnp')],[]).
rule(acquisition, acquired, 3, 0,
     [item(_,[',','vbg']),item(_,['nnp','nn']),item(_,['cd','.']),item(_,_)],
     [list(2,_,'nnp')],[item(['{','said'],_),item(['ncro.l','the'],_),item(_,[')','nn']),
     list(2,_,_),item(_,'nn')]).
rule(acquisition, acquired, 4, 0,
     [list(2,['to','of','acquire','-'],_)],[item(_,'nnp'),
     item(['inn','hudson','semiconductor'],'nnp'),item(_,'nnp')],[]).
rule(acquisition, acquired, 4, 0,
     [item(['394','of','the'],_)],[item(['piedmont','bankamerica','antonson'],'nnp'),
     item(_,'nnp'),item(_,'nnp')],[]).
rule(acquisition, acquired, 3, 0,
     [item(_,['nn','in']),item(_,[',','vbg'])],[list(2,_,['jj','nnp']),
     item(_,_),item(['international','inc'],'nnp')],[item(_,[',','(']),
     item(['and','wen'],_)]).
rule(acquisition, acquired, 4, 0,
     [],[item(['moore','dome','u.s.'],'nnp'),list(3,_,['cc','nn','nnp']),
     item(['inc','ltd','parts'],_)],[item(_,['(','nnp','nn'])]).
rule(acquisition, acquired, 4, 0,
     [],[item(['retail','scandinavia','industrial'],_),list(3,_,_),
     item(['inc','division'],_)],[]).
rule(acquisition, acquired, 3, 0,
     [],[item(['conrac','monier'],'nnp'),item(_,'nnp')],[]).
rule(acquisition, acquired, 3, 0,
     [item(_,['nns','in']),item(_,['in','('])],[list(2,_,'nnp'),
     item(['holding','industries'],'nnp'),item(['corp','inc'],'nnp')],[]).
rule(acquisition, acquired, 3, 0,
     [],[item(_,'nnp'),item(_,'nnp'),item(_,_),list(2,_,_),
     item(['association','chimie'],'nnp')],[item(_,['(','to'])]).
rule(acquisition, acquired, 3, 0,
     [],[item(_,_),item(['corp','corn'],_),list(2,_,_),item(['america','business'],_)],
     []).
rule(acquisition, acquired, 3, 0,
     [item(_,[')','('])],[list(3,_,['nn','nnp']),item(['business','inc'],_)],
     [item(_,['.',')']),item(_,_),item(_,['nns','vbd']),list(2,_,_),item(_,['dt','nnp'])]).
rule(acquisition, acquired, 3, 0,
     [],[item(['state','national'],_),list(2,['of','television','bank'],_),
     item(_,['nn','nnp']),item(_,'nnp')],[item(',',','),list(2,_,_),
     item(_,['nn','('])]).
```

```
rule(acquisition, acquired, 3, 0,
     [],[item(['interlake','central'],'nnp'),item(_,'nnp'),
     list(3,['buckhannon','of','co','bank'],_)],[item(_,['cc',','])]).
rule(acquisition, acquired, 4, 0,
     [item(_,['in','('])],[item(_,'nnp'),item(['bankshares','savings','pantries'],_),
     item(_,'nnp')],[]).
rule(acquisition, acquired, 3, 0,
     [item(_,['vbg','vb'])],[item(_,_),item(['television','industries'],['nn','nnp']),
     item(_,['nn','nnp'])],[]).
rule(acquisition, acquired, 3, 0,
     [],[item(_,['dt','nnp']),item(_,['nn','nnp']),item(_,['in','nnp']),
     list(2,_,_),item(['township','association'],'nnp')],[item(_,'in')]).
rule(acquisition, acquired, 3, 0,
     [item(_,['prp$','vb'])],[list(3,_,'nnp'),item(['abex','co'],'nnp')],
     [item(_,[',','in'])]).
rule(acquisition, acquired, 3, 0,
     [item(['9','share'],_),item(_,[':','in'])],[list(2,_,'nnp'),
     item(['industries','products'],'nnp'),item(['inc.','inc'],'nnp')],[]).
rule(acquisition, acquired, 6, 0,
     [item(_,['vb','vbn']),item(_,['(','to'])],[item(_,'nnp'),item(_,_),item(_,'nnp'),
     item(['co','inc'],'nnp')],[item(_,[')','cc'])]).
rule(acquisition, acquired, 3, 0,
     [],[item(['rb','allegheny'],'nnp'),item(_,'nnp'),item(['inc','co'],'nnp')],
     [list(2,_,_),item(_,['in','nn'])]).
rule(acquisition, acquired, 4, 0,
     [item(['in','producer'],_)],[list(2,_,'nnp'),item(['inc.','ltd'],'nnp')],[]).
rule(acquisition, acquired, 4, 0,
     [],[item(['dey','first','merchants'],'nnp'),list(6,_,_),
     item(['stores','hohenwald','park'],'nnp')],[]).
rule(acquisition, acquired, 3, 0,
     [item(_,['prp$',':'])],[list(2,_,'nnp'),item(['world','equipment'],'nnp'),
     item(['inc','co'],'nnp')],[]).
rule(acquisition, acquired, 4, 0,
     [item(['acquire','unit'],_),item(_,['(','vbz'])],[item(_,'nnp'),
     list(2,_,_),item(['tv','inc'],_)],[item(_,[')','nn'])]).
rule(acquisition, acquired, 3, 0,
     [list(2,_,_),list(2,_,_),item(['of','''s'],_)],[item(_,'nnp'),
     item(['financial','manufacturing'],'nnp'),item(_,'nnp')],[]).
rule(acquisition, acquired, 4, 0,
     [item(_,['in','jj'])],[list(3,_,'nnp'),item(['inc','inc.'],'nnp')],
     [item(['gulf','common'],_)]).
rule(acquisition, acquired, 3, 0,
     [item(_,['(',','])],[list(2,_,'nnp'),item(['lamborghini','systems'],'nnp'),
     item(_,'nnp')],[item(_,[')',','])]).
rule(acquisition, acquired, 3, 0,
     [item(_,['cd','nn']),item(_,[':','in'])],[list(3,['southern','fe','hughes',
     'santa'],'nnp'),item(_,'nnp'),item(['co','corp'],'nnp')],
     [item(['rose',','],_),item(_,['cd','dt']),item(_,['to','nn'])]).
rule(acquisition, acquired, 7, 0,
     [item('shares','nns'),item('of','in')],[item(_,'nnp'),
     list(2,_,_),item('inc','nnp')],[]).
rule(acquisition, acquired, 3, 0,
     [],[item(['revlon','trilogy'],'nnp'),item(_,'nnp'),item(_,'nnp')],[]).
rule(acquisition, acquired, 3, 0,
     [],[item(['taft','kresge'],'nnp'),item(_,['nnp','nn']),item(_,['nnp','nns'])],
     []).
rule(acquisition, acquired, 3, 0,
     [item(_,['nn','in']),item(['in','two'],_)],[item(_,'nnp'),item(_,'nnp'),
     item(['corp','blocks'],_)],[]).
rule(acquisition, acquired, 3, 0,
     [],[item(['westfiar','imperial'],'nnp'),list(2,_,'nnp'),
     item(['corp','association'],'nnp')],[]).
rule(acquisition, acquired, 3, 0,
     [item(_,['vbd','in']),list(2,_,_)],[item(_,'nnp'),list(2,_,'nnp'),
     item(['physik','property'],'nnp')],[]).
rule(acquisition, acqloc, 3, 0,
     [item(['firm','}'],_),item(['based',','],_),item(['in','a'],_)],
     [list(3,_,[',','nnp'])],[item(_,['.',':']),item(_,['endsent','vbn'])]).
rule(acquisition, acqloc, 3, 0,
     [],[item(_,'nnp'),list(2,_,[',','cc','nnp']),
     item(['washington','california'],'nnp')],[]).
rule(acquisition, acqloc, 3, 0,
     [item([',','co'],_),item(['based','}'],_),item(_,'in')],[item(_,'nnp'),
     item(',',','),list(2,_,['.','nnp'])],[item(_,['.',',']),
     list(2,_,_),item(_,['vbd','jj']),list(2,_,_),item(_,['in','vbg'])]).
rule(acquisition, acqloc, 5, 0,
     [item(_,['nn','in']),list(2,_,_),item(['in','of'],'in')],[item(_,'nnp'),
     item(',',','),list(5,_,_)],[item(_,['cc',',']),item(['the','and'],_),list(2,_,_)]).
rule(acquisition, acqloc, 3, 0,
     [item(_,['nn','nns']),item('in','in')],[item(_,'nnp'),item(',',','),item(_,'nnp')],
     [item(',',','),item(_,['cc','in'])]).
rule(acquisition, acqloc, 4, 0,
     [item(['based','tampa','l.'],_),item(_,_)],[list(2,_,_),
     item(['n.j.','petersburg','ontario'],'nnp')],[]).
rule(acquisition, acqloc, 5, 0,
     [item(['the','a','in','an'],_)],[list(3,_,['nnp',',','rb']),
     item(['pacific','diego','germany','n.y.'],'nnp')],[item(_,['to','nn',',','vbg'])]).
rule(acquisition, acqloc, 8, 0,
     [item(_,['in','prp$'])],[item(['italian','french','canada','michigan'],_)],[]).
rule(acquisition, acqloc, 4, 0,
     [item(['extensive','a','the'],_)],[list(3,_,['cc',',','jj','nnp']),
     item(['.','canadian','seattle'],_)],[item(_,['nn',':','nns'])]).
rule(acquisition, acqloc, 3, 0,
     [item(_,['.',',']),item(_,_),item(['western','in'],['nnp','in'])],
     [list(3,_,['.','nnp'])],[item(['is',','],_),list(2,['gas','and','a'],_),
     item(_,['vbz','nn'])]).
rule(acquisition, acqloc, 4, 0,
     [item(_,['vb','in',','])],[item(['cleveland','london','brazil'],'nnp')],
     [item(_,['nn',','])]).
rule(acquisition, acqloc, 3, 0,
     [item(_,['dt','prp$'])],[item(_,_),item(['angeles','german'],_)],[item(_,[':','nn'])]).
rule(acquisition, acqloc, 3, 0,
     [item(_,['in','jj'])],[list(2,_,[',','nnp']),item(['jersery','n.y.'],'nnp')],[]).
rule(acquisition, acqloc, 3, 0,
     [item('in','in')],[list(2,_,[',','nnp']),item(['kong','ohio'],'nnp')],[]).
rule(acquisition, acqloc, 3, 0,
     [item(_,'nn'),item(_,['to','nns']),item(_,_),item(_,['vbg','jj'])],[item(_,'nnp')],
     [item(_,['nns','cc']),item(_,_),list(2,_,_),item(_,['vbn','vbd'])]).
rule(acquisition, acqloc, 3, 0,
     [item(_,['to','in']),list(2,['endsent','central','printer'],_)],
     [item(['kansas','toronto'],'nnp')],[item(['utility',','],_)]).
rule(acquisition, acqloc, 3, 0,
     [item(['in','of'],'in')],[item(['w.','new'],'nnp'),item(_,'nnp')],[item(_,_),
     item(_,['endsent','nnp'])]).
rule(acquisition, acqloc, 3, 0,
     [],[item(['south','monterey'],_),list(3,['calif',',','east','park'],_),
    item(_,['nnp','.'])],[item(_,_),item(_,['vbp','endsent'])]).
rule(acquisition, acqcode, 4, 0,
     [item(['piedmont','petroleum','hughes'],'nnp'),list(2,_,'nnp'),item('{','(')],
     [item(['pie','dmp','ht'],'nnp')],[item('}',')'),list(1,_,_),item(_,['in','endsent']),
     list(2,_,_),item(_,['nn',','])]).
rule(acquisition, acqcode, 23, 1,
     [item(_,['jjr','('])],[item(_,'nnp')],[item(_,['jj',')']),
     item(['holdings','shares','stake','founder'],_)]).
```

```
rule(acquisition, acqcode, 4, 0,
     [item(_,_),list(2,_,'nnp'),item(_,'nnp'),item('{','(')],
     [item(['dmp.mo','gy','tfb'],'nnp')],[]).
rule(acquisition, acqcode, 52, 2,
     [],[item(['bko','amdc','fen','rtb','eye','cax','gnva','tc','caw','unicoa','aix','rni',
     'csbk','dca','mnra.s','ncro.l','trmw','crtr.o','kwp','pvdc','atpi','cds','rev','symb',
     'cyl','hay','mtx','fin','dnam','clev','gaf','pza','wen','biod','valt','rpch','efh',
     'icgs.l','prsl','inet','flt','fcsi'],'nnp')],[]).
rule(acquisition, acqcode, 12, 0,
     [item(_,['(','to'])],[item(['pepi.o','pnp','rhml.l','gsti','ag','blas.o','sgl','npt',
     'fte','datr','efac'],'nnp')],[]).
rule(acquisition, acqcode, 13, 0,
     [item(_,'nnp'),list(2,_,_),item(_,'nnp'),item('{','(')],[item(_,'nnp')],
     [item('}',')'),item(['stake','call'],_)]).
rule(acquisition, acqcode, 3, 0,
     [],[item(['enzn','u'],'nnp')],[item('}',')'),item(_,_),item(_,['nnp','dt']),
     item(_,[',','nnp'])]).
rule(acquisition, acqcode, 7, 0,
     [item(_,['nns','nnp']),item('{','(')],[item(_,'nnp')],[item('}',')'),
     item(['stake','endsent'],_),item(['for','washington'],_)]).
rule(acquisition, acqcode, 3, 0,
     [item(['fund','metrobanc'],'nnp'),item('{','(')],[item(_,'nnp')],
     [item('}',')')]).
rule(acquisition, acqbus, 4, 0,
     [item(_,['in','vbz','nnp'])],[item(_,'nn'),list(2,_,_),item(_,'nn'),
     item(['systems','development'],_)],[]).
rule(acquisition, acqbus, 4, 0,
     [],[item(_,['nn','jj']),item(_,'nn'),item(['rental','centers','equipment'],_)],
     []).
rule(acquisition, acqbus, 3, 0,
     [item(_,['nnp','rb']),list(2,_,_),item(['wholesales','the'],_)],
     [list(2,['and','nicotine','parts'],_),item(_,['nn','nns'])],
     [item(['to','market'],_),list(2,_,_),item(_,['jj','prp'])]).
rule(acquisition, acqbus, 5, 0,
     [list(2,['has','of','massive','finlays','a'],_)],
     [item(['confectionery','oil','hermetic','federal'],['nn','jj']),list(3,_,_),
     item(['newsagent','gas','packages','bank'],_)],[]).
rule(acquisition, acqbus, 4, 0,
     [item(_,'dt')],[item(_,'nn'),item(_,['nns','nn'])],[item(['unit','company'],'nn')]).
rule(acquisition, acqbus, 4, 0,
     [item(['its','of','italian'],_)],[list(3,_,['vbn','nn','jj']),
     item(['circuits','products','maker'],_)],[item(_,_),item(_,['nnp',','])]).
rule(acquisition, acqbus, 3, 0,
     [item(_,['nn',',']),list(2,_,_),item(_,_),item(_,['rb','nnp']),list(2,_,_),
     item(['affiliated','the'],_)],[list(2,_,['jj','nn'])],[item(['companies','had'],_),
     item(_,_),item(_,['dt','nn'])]).
rule(acquisition, acqbus, 3, 0,
     [item(_,['nn','vb']),item(_,_),item(['seven','-'],_),item(_,['jj','vbn'])],
     [list(2,_,_)],[item(['and','maker'],_)]).
rule(acquisition, acqbus, 5, 0,
     [item(['ocean','distributes','term','makes'],_)],[list(3,_,[':','nn','jj']),
     item(_,['vbg','nns'])],[item(_,_),item(_,['to','endsent','dt'])]).
rule(acquisition, acqbus, 3, 0,
     [item(_,['nn','endsent']),item(_,_),item(_,_),item(_,_),item(['superior','makes'],_)],
     [list(8,_,_)],[item(['accounts',','],_),list(2,_,_)]).
rule(acquisition, acqbus, 3, 0,
     [],[item(['travel','leasing'],'nn'),list(3,_,_),item(_,_)],
     [item([',','system'],_),list(2,_,_)]).
rule(acquisition, acqbus, 3, 0,
     [item(_,['dt','in'])],[item(_,_),item(['abatement','products'],_)],
     [item(_,['nn',','])]).
rule(acquisition, acqbus, 5, 0,
         [item(_,['cc','to','nn','nnp']),item(_,_),item(['of','its','in','a'],_)],
         [list(2,_,['jj','nn']),item(['interconnect','products','instruments','merchandise'],_)],
         [list(2,_,_),item(_,['wdt','nns','to','nn'])]).
    rule(acquisition, acqbus, 3, 0,
         [item(['maker','the'],_),item(_,['in','jjs'])],[item(_,['vbn','nn']),
         item(_,['nns','nn'])],[]).
    rule(acquisition, acqbus, 4, 0,
         [item(['only','of','two'],_)],[list(2,_,_),
         item(['gold','holding','ferrosilicon'],['vbg','nn'])],[item(_,'nns')]).
    rule(acquisition, acqbus, 3, 0,
         [item(_,['nn','nnp']),item(_,_),item(['markets','provides'],_)],[list(10,_,_)],
         [item(_,[',','to']),list(2,_,['in','nnp','nn']),item(_,['vbd','nnp']),list(2,_,_)]).
    rule(acquisition, acqbus, 3, 0,
         [item(_,['nn',',']),list(2,_,_)],[item(_,'nn'),item(_,'nn')],
         [item(['facility','firm'],'nn')]).
    rule(acquisition, acqbus, 3, 0,
         [item(_,['md','jj']),item(_,_),item(['in','and'],_),item(['the','seven'],_)],
         [item(_,['jj','nn']),item(_,'nn')],[item(_,['nn','nns'])]).
    rule(acquisition, acqbus, 3, 0,
         [item(_,['nnp','vbn']),item(_,['vbz','in'])],[list(3,_,'nn'),
         item('and','cc'),item(_,'nn'),item(_,['nns','nn'])],[]).
    rule(acquisition, acqbus, 3, 0,
         [item(_,['nnp','nn']),list(2,['60',',','includes'],_),item(['operating','its'],_)],
         [list(3,_,_)],[item(['in','company'],_),list(2,_,_),list(3,_,_),item(_,['md','in'])]).
    rule(acquisition, acqbus, 3, 0,
         [item(_,['vbz','cd'])],[item(['community','accounting'],'nn'),list(2,_,['nns','nn'])],
         [item(_,['to','in'])]).
    rule(acquisition, acqbus, 4, 0,
         [item(['its','ten'],_)],[list(4,_,_)],[item(['unit','restaurants'],_),
         item(_,[',','in']),item(_,['nnp','jj']),item(_,'nnp')]).
    rule(acquisition, acqbus, 3, 0,
         [item(_,['vbz','vbn']),list(2,_,_)],[item(['auto','convenience'],'nn'),
         item(_,'nns')],[]).
    rule(acquisition, acqbus, 3, 0,
         [item(_,['cd','dt']),item(_,_),list(2,['company','stg','held'],_),item(_,_)],
         [item(['supermarket','video'],'nn'),list(2,_,_)],[item(_,['in','nn']),
         item(_,_),item(_,['cc','nnp'])]).
    rule(acquisition, acqbus, 3, 0,
         [item(['owned','concrete'],_)],[item(_,'nn'),item(_,'nn')],[item(_,['in','nn'])]).
    rule(acquisition, acqbus, 3, 0,
         [item(_,['nn','nnp']),item(_,'vbz')],[item(_,'nn'),item(_,'nns')],
         [item(_,_),item(_,_),item(_,['cd','nnp'])]).
    rule(acquisition, acqbus, 3, 0,
         [item(['a','cenergy'],_),list(2,_,_),list(2,['a','owned',','],_),
         list(2,['of','u.s.','manufacturer'],_)],[list(2,['and','colorants','oil'],_),
         item(_,['in','nn']),item(_,['dt','nn']),item(_,['nns','cc']),item(_,'nn')],
         [item(_,['.','nn']),item(_,_),list(2,_,_),item(['hanna','said'],_),list(2,_,_)]).
    rule(acquisition, acqbus, 3, 0,
         [item(_,['nn','jj'])],[item(['oil','feedstuff'],'nn'),item(_,_),item(_,'nn')],
         [item(_,['nn','nns'])]).
    rule(acquisition, acqbus, 3, 0,
         [item(['in',','],_),item(['toronto','an'],_),list(2,_,_)],[item(['oil','collects'],_),
         list(3,['oils','lubricating','products','used'],_)],[item(_,['in','nn'])]).
    rule(acquisition, acqbus, 3, 0,
         [item(_,['nnp',',']),item(_,_),item(_,[')','vbz'])],[list(3,_,[':','nn','vbn']),
         item(_,['nn','nns'])],[item(_,['.',',']),item(['endsent','had'],_)]).
    rule(acquisition, acqbus, 3, 0,
         [],[item(['hotel','brokerage'],_)],[item(_,['in','nns'])]).
    rule(acquisition, acqbus, 5, 0,
         [item(_,['vbn','vbz'])],[list(3,_,['vbg',':','nns','nn']),item('and','cc'),item(_,_),
         item(_,'nns')],[]).
    rule(acquisition, acqabr, 4, 0,
      [item(_,['nn','nnp']),item(_,['pos','vbz','in'])],[item(_,'nnp'),
      list(2,_,['nn','nnp',':'])],[item(['tobacco','sale','lithographing'],_)]).
rule(acquisition, acqabr, 4, 0,
      [item(['west','march','to'],_),item(_,_),item(_,['vbz',':','jj'])],[item(_,'nnp'),
      list(2,['bank','crellin','tool','-'],_)],[item(['endsent','co','shares'],_),
      item(_,_),item(_,[',','cd','jj'])]).
rule(acquisition, acqabr, 3, 0,
      [item(_,['vb','vbd','prp$']),list(2,_,_),list(2,_,_)],
      [item(['american','builders','gerber'],'nnp'),item(_,'nnp')],[]).
rule(acquisition, acqabr, 3, 0,
      [item(_,['vbd','dt']),item(_,['prp$','nn']),item(['stake','of'],_),item(_,['in','dt'])],
      [item(_,'nnp'),item(_,'nnp')],[item(_,'nnp'),item(_,['nnp','in'])]).
rule(acquisition, acqabr, 5, 0,
      [item(_,['vbn','nnp']),item(_,['cc','nn',')']),item(['{','buys','of','purchases'],_)],
      [item(_,['nns','nnp'])],[item(['inc','{','aviation','endsent'],_)]).
rule(acquisition, acqabr, 7, 0,
      [item(_,['cd','(','nn','vbd'])],[item(['cyclops','western','europeesche','unitek'],_)],
      []).
rule(acquisition, acqabr, 3, 0,
      [item(_,['nnp','nn','prp$']),item(_,['pos','vbz','jj'])],[list(3,_,'nnp')],
      [item(['tobacco','sale','industries'],_)]).
rule(acquisition, acqabr, 3, 0,
      [item(_,['nnp','nns']),item(['craft','to'],_),item(['(','buy'],_)],[list(2,_,'nnp')],
      [item(_,['sym','endsent']),item(_,_),item(_,['endsent',','])]).
rule(acquisition, acqabr, 6, 0,
      [],[item(['brinkmann','revlon','rospatch','wyona','plainwell'],'nnp')],[]).
rule(acquisition, acqabr, 4, 0,
      [],[item(['hayes','santa','south'],'nnp'),list(3,_,_),
      item(['albion','southern','l'],'nnp')],[item(_,_),item(_,['vbd','dt','nnp'])]).
rule(acquisition, acqabr, 4, 0,
      [item(_,['nn','jj','dt']),item(['that','boosts','13.3'],_)],[list(3,_,['cc','nnp'])],
      [item(['petroleum','{','common'],_),list(2,_,_),item(_,_),item(_,['endsent','nnp'])]).
rule(acquisition, acqabr, 15, 0,
      [item(['5.1',',','interest','bid'],_),item(_,['in','cd','jj'])],[item(_,'nnp')],
      [item(['shares','common','mining','corp'],_),item(_,['cc','nns','nnp',','])]).
rule(acquisition, acqabr, 5, 0,
      [],[item(['fermenta','fairchild','sgl','coastal'],'nnp')],[]).
rule(acquisition, acqabr, 4, 0,
      [list(2,['in','pct','.','93'],_),item(['{','of','endsent'],_)],[item(_,['nnps','nnp'])],
      [item(['computer','''','is'],_),item(['associates','voting','the'],_)]).
rule(acquisition, acqabr, 3, 0,
      [item(['interest','pct'],'nn'),item(['in','of'],'in')],[item(_,'nnp'),item(_,_)],
      [item(['inc','{'],_)]).
rule(acquisition, acqabr, 3, 0,
      [item(_,['vbg','('])],[item(_,'nnp'),list(2,['poulenc','engineering','-'],_)],
      [item(['and','chimie'],_),item(_,_),item(['inc','}'],_)]).
rule(acquisition, acqabr, 4, 0,
      [item(_,['jj','nns']),item(_,_),item(['in','endsent'],_)],[list(2,_,'nnp')],
      [item(['electric','is'],_),item(['products','a'],_)]).
rule(acquisition, acqabr, 4, 0,
      [],[item(['cadillac','radiation','moore'],_),item(_,'nnp')],[]).
rule(acquisition, acqabr, 3, 0,
      [item(_,['nns','rb']),list(2,_,_),item(_,['prp$','endsent'])],[item(_,_),
      item(['pafific','and'],_),item(_,'nnp')],[item(_,['nn',','])]).
rule(acquisition, acqabr, 3, 0,
      [list(2,_,_),item(_,['nnp','vbd']),item(_,_),item(_,[':','endsent'])],
      [item(['progressive','pay'],_),list(3,_,['nnp','pos'])],[item(_,['nnp','vbz']),
      item(_,_),item(_,_),item(_,['prp','in'])]).
rule(acquisition, acqabr, 3, 0,
      [item(_,['in','('])],[item(['computer','allegheny'],'nnp'),list(2,_,['nn','nnp'])],
      [item(['{','systems'],_)]).
rule(acquisition, acqabr, 4, 0,
     [item(_,['nnp','jj']),item(_,['nnps','nn']),item(['to','of'],_)],
     [list(2,['''s','service','debbie'],_),item(_,'nnp')],[item(_,['.','cc']),
     item(_,_),item(_,['nnp','nns'])]).
rule(acquisition, acqabr, 5, 0,
     [],[item(['becher','dei','metex','zoran'],'nnp')],[]).
rule(acquisition, acqabr, 6, 0,
     [item(['said','based','of','{','which'],_)],
     [item(['hermetronics','tricil','gencorp','temco','hutton'],'nnp')],[]).
rule(acquisition, acqabr, 4, 0,
     [item(['pct','letter'],'nn'),item(_,['in','to'])],[item(_,'nnp')],
     [item('''s','pos'),list(2,_,_),item(_,['nns','dt'])]).
rule(acquisition, acqabr, 3, 0,
     [item(['endsent','says'],_)],[item(_,'nnp'),item(_,['nnp','nn'])],
     [item(['provides','sale'],_)]).
rule(acquisition, acqabr, 3, 0,
     [item(['sells','{'],_)],[item(_,'nnp')],[item(_,['nn','cc']),
     item(_,['in','nnp']),item(['16.9','inc'],_)]).
rule(acquisition, acqabr, 4, 0,
     [item(['station','acquire'],_)],[item(_,'nnp')],[item(_,_),item(_,['vbg','nnp']),
     item(_,_),item(_,'nnp'),item(['nationale',','],_)]).
rule(acquisition, acqabr, 4, 0,
     [],[item(['conrac','kdi','spectrum'],'nnp')],[]).
rule(acquisition, acqabr, 8, 0,
     [item(_,['nnp','to','in']),item(_,_),item(_,_),item(['equity','stake','ltd'],_),
     item(['in','{'],_)],[list(2,_,[')','nnp'])],[item(_,['endsent','to','in'])]).
rule(acquisition, acqabr, 6, 0,
     [],[item(['lamborghini','cenergy','linotype'],'nnp')],[]).
rule(acquisition, acqabr, 10, 0,
     [item(['company','qintex','pct'],_),item(['said','extends','of'],_)],[item(_,'nnp')],
     [list(2,_,_),item(['f.','}','common'],_)]).
rule(acquisition, acqabr, 3, 0,
     [item(_,['in',':'])],[item(_,'nnp')],[item(['enterprises','development'],_),
     list(2,_,_),item(_,['in','prp$'])]).
rule(acquisition, acqabr, 4, 0,
     [item(_,['to','nn']),item(['acquire','sells'],_)],[item(_,['nns','nnp']),
     item(_,['nnp','nn'])],[item(['sacramento','endsent'],_)]).
rule(acquisition, acqabr, 3, 0,
     [item(_,['nn','vbn']),item(_,['nn','vbg']),list(2,_,_)],[item(['dome','dayton'],'nnp'),
     item(_,'nnp')],[]).
rule(acquisition, acqabr, 3, 0,
     [item(_,['nnp','nn']),item(_,_),item(['''s','{'],_)],[list(2,_,['nnps','nnp'])],
     [item(['business','stores'],_)]).
rule(acquisition, acqabr, 3, 0,
     [item(_,['nnp','in']),list(2,_,_),item(_,'in')],[item(_,'nnp')],
    [item(['should','inn'],_)]).
rule(acquisition, acqabr, 3, 0,
     [item(_,['nn','vbd']),item(_,['nn','.']),item(['at','endsent'],_)],
     [item(_,'nnp'),item(_,'nnp')],[item(_,['cc','md']),item(_,_),item(_,['vb','in'])]).
rule(acquisition, acqabr, 3, 0,
     [item(_,['in','vbd']),item(_,['vbg','('])],[item(_,'nnp')],
     [item(['''s','technische'],_)]).
rule(acquisition, acqabr, 3, 0,
     [],[item(_,_),item(['toll','world'],'nn')],[item(_,['nnp','('])]).
rule(acquisition, acqabr, 3, 0,
     [item(['said','of'],_)],[list(2,_,'nnp')],[item(['operates','resources'],_)]).
rule(acquisition, acqabr, 3, 0,
     [],[item(['supermarkets','scan'],'nnp'),list(2,['graphics','general','-'],_)],
     [list(2,_,['vb','vbz','to']),item(_,['dt','in'])]).
rule(acquisition, acqabr, 3, 0,
     [item(_,['nn','vbz']),item(_,['nn','nnp']),item('for','in')],[item(_,'nnp')],
     [item([',','merger'],_)]).
rule(acquisition, acqabr, 3, 0,
     [item(['houston','buy'],_)],[item(_,'nnp')],[item(['for','inc.'],_)]).
rule(acquisition, acqabr, 9, 0,
     [item(_,['to',')','nn']),item(['purchase','has','of'],_)],[item(_,'nnp')],
     [item(['consolidated','{','stock'],_)]).
rule(acquisition, acqabr, 9, 0,
     [],[item(['rhm','cyclops','cgct','uab'],'nnp')],[]).
rule(acquisition, acqabr, 3, 0,
     [item(_,[',','nns']),list(2,_,_),item(['12','for'],_),item(_,[':','dt'])],
     [item(_,'nnp')],[item(['corp','share'],_),list(2,_,_),item(_,['prp$','nnp'])]).
rule(acquisition, acqabr, 3, 0,
     [item(['in','sells'],_)],[item(['shearson','national'],'nnp'),item(_,'nnp')],[]).
rule(acquisition, acqabr, 5, 0,
     [],[item(['pizza','ic','trilogy'],'nnp'),item(['inn','gas','resource'],'nnp')],[]).
rule(acquisition, acqabr, 3, 0,
     [item(_,[',','dt']),item(['each','four'],_)],[item(_,'nnp'),item(_,['nnp','nnps'])],[]).
rule(acquisition, acqabr, 3, 0,
     [item(['for','buy'],_)],[item(['nichols','usair'],'nnp')],[]).
rule(acquisition, acqabr, 3, 0,
     [item(_,['endsent','nnp']),item(_,_),item(_,['vbd','endsent'])],[item(_,'nnp'),
     item(_,['nnps','nnp'])],[item(['has','had'],_)]).
rule(acquisition, acqabr, 4, 0,
     [item(['endsent','paks'],_),list(2,_,_),item(['said','buy'],_)],[item(_,'nnp')],
     [item(_,['nnps','nn'])]).
rule(acquisition, acqabr, 4, 0,
     [item(['holdings','acquisition'],_),item(['in','of'],'in')],[list(2,_,'nnp')],
     [item(['.','inc'],_),item(_,['endsent','in'])]).
rule(acquisition, acqabr, 4, 0,
     [item(_,['vbd','endsent'])],[item(_,'nnp')],[item(['general','had'],_),
     list(2,_,_),item(_,['(','in'])]).
rule(acquisition, acqabr, 3, 0,
     [item(['394','for'],_)],[item(_,'nnp')],[item(_,['nns','pos'])]).
rule(acquisition, acqabr, 3, 0,
     [item(_,['prp$','pos']),item(['dome','said'],_)],[list(2,_,['nnps','nnp'])],
     [item(_,['pos','vbd'])]).
rule(acquisition, acqabr, 5, 0,
     [item(['its','march'],_),item(['stake','18'],_),item(['in','-'],_)],
     [list(2,_,'nnp')],[item(['ltd','inc'],'nnp')]).
rule(acquisition, acqabr, 3, 0,
     [],[item(['norddeutsche','foote'],'nnp'),item(_,'nnp')],[]).
rule(acquisition, acqabr, 4, 0,
     [item(['hold','principle'],_),item(['in','to'],_),item(['{','acquire'],_)],
     [item(_,'nnp'),list(2,_,'nnp')],[item(_,'nnp'),item(_,[')','in'])]).
rule(acquisition, acqabr, 3, 0,
     [item(['remainder','stake'],'nn'),item(_,'in')],[item(_,'nnp')],
     [item(_,['nns','endsent']),list(2,_,_),item(['cheaper','march'],_)]).
rule(acquisition, acqabr, 3, 0,
     [item(_,[')','nnp']),item(['to','pincus'],_),item(['buy','starts'],_)],
     [list(3,_,['in','nnp'])],[item(_,['endsent','(']),item(_,'nnp'),item(_,[',',')']),
list(2,['va','bid','w.'],_),item(_,['endsent','.'])]).
rule(acquisition, acqabr, 3, 0,
     [item(_,['vbd','vb']),list(2,['dealings','columbus','its'],_),list(2,_,_)],
     [item(['builders','american'],'nnp'),item(_,'nnp')],[]).
rule(acquisition, acqabr, 3, 0,
     [item(_,['vbn','nn']),item(['by','said'],_)],[list(2,_,'nnp')],
     [item(['newspapers','shareholders'],_),list(2,_,_),list(2,_,_),item(_,['vbz','cd'])]).
rule(acquisition, acqabr, 7, 0,
     [item(['in','its'],_)],[item(_,'nnp')],[item(['{','corp'],_)]).
rule(acquisition, acqabr, 3, 0,
     [item(_,[',','dt']),item(['1986','board'],_),item([',','of'],_)],[item(_,'nnp')],
     [item(_,[':','nnp'])]).
rule(acquisition, acqabr, 5, 0,
     [],[item(['taft','caesars'],'nnp'),item(_,'nnp')],[]).
```

```
rule(acquisition, acqabr, 3, 0,
     [item(['buys','in'],_)],[item(['antonson','shearson'],'nnp')],[]).
rule(acquisition, acqabr, 3, 0,
     [item(['said','to'],_),item(['that','acquire'],_)],[list(2,_,['nn','nnp'])],
     [item(_,[',','endsent']),item(_,['vbn','nnp']),item(_,['in','nnp'])]).
rule(acquisition, acqabr, 3, 0,
     [],[item(['horizon','consolidated'],'nnp')],[item(_,['in','jj'])]).
rule(acquisition, acqabr, 3, 0,
     [item(_,['cd','nnp']),list(2,_,_),item(_,_),item(_,['vbn','.']),list(2,_,_),
     item(_,['vbd','dt'])],[item(_,'nnp')],[item(['cement','board'],_)]).
```

Appendix C

Part-of-Speech Tags Used

This appendix contains a list of the part-of-speech tags used in the experiments described in Chapter 4.

TAG	Meaning
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential "there"
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PP	Personal pronoun
PP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	"to"
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle

VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Wh-determiner
WP	Wh-pronoun
WP$	Possessive wh-pronoun
WRB	Wh-adverb

Bibliography

Aone, 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 Linguistics*, pp. 122–129 Cambridge, MA.

Beeferman, D., Berger, A., & Lafferty, J. (1997). Text segmentation using exponential models. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pp. 35–46.

Bennett, S., Aone, C., & Lovell, C. (1997). Learning to tag multilingual texts through observation. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pp. 109–116.

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

Breiman, L. (1998). Bagging predictors. *Machine Learning*, *24*(2), 123–140.

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*, pp. 722–727.

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.

Cardie, C. (1997). Empirical methods in information extraction. *AI Magazine*, *18*(4), 65–80.

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 Spring Symposium on Machine Learning in Information Access*, pp. 18–25. AAAI Press.

Cohen, W. (1995). Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning.*

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.

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

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

DARPA (Ed.). (1995). *Proceedings of the 6th Message Understanding Conference*, San Mateo, CA. Morgan Kaufman.

De Raedt, L., & Bruynooghe, M. (1993). A theory of clausal discovery. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pp. 1058–1063 Chambery, France.

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 Linguistics* Santa Cruz, CA.

Fellbaum, C. (Ed.). (1998). *WordNet: An Electronic Lexical Database.* MIT Press, Cambridge, MA.

Fisher, D., Soderland, S., McCarthy, J., Feng, F., & Lehnert, W. (1995). Description of the UMass systems as used for MUC-6. In *Proceedings of the 6th Message Understanding Conference*, pp. 127–140.

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

Freitag, D. (1997). Using grammatical inference to improve precision in information extraction. In *Working Notes of the ICML-97 Workshop on Automata Induction, Grammatical Inference, and Language Acquisition.*

Freitag, D. (1998a). Information extraction from HTML: Application of a general learning approach. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence.*

Freitag, D. (1998b). Multi-strategy learning for information extraction. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pp. 161–169.

Freitag, D. (1998c). Toward general-purpose learning for information extraction. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics.*

Freund, Y., & Schapire, R. (1996). Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pp. 148–156.

Freund, Y., & Schapire, R. E. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. Technical report, AT&T Bell Laboratories, Murray Hill, NJ.

Holte, R. C., Acker, L., & Porter, B. (1989). Concept learning and the problem of small disjuncts. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp. 813–818.

Hsu, C.-N., & Dung, N.-T. (1998). Wrapping semistructured web pages with finite-state transducers. In *Proceedings of the Conference on Automated Learning and Discovery.*

Huffman, S. B. (1996). Learning information extraction patterns from examples. In Wermter, S., Riloff, E., & Scheler, G. (Eds.), *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*, pp. 246–260. Springer, Berlin.

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

Kushmerick, N., Weld, K., & Roorenbos, R. (1997). Wrapper induction for information extraction. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence.*

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

Lehnert, W., McCarthy, J., Soderland, S., Riloff, E., Cardie, C., Peterson, J., Feng, F., Dolan, C., & Goldman, S. (1993). UMASS/HUGHES: Description of the CIRCUS system used for MUC-5. In *Proceedings of the Fifth Message Understanding Conference*, pp. 277–291.

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.

Liere, R., & Tadepalli, P. (1997). Active learning with committees for text categorization. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pp. 591–596.

Magerman, D. M. (1995). Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, 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.

Michalski, R. S. (1983). A theory and methodology of inductive learning. *Artificial Intelligence*, *20*, 111–161.

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 Linguistics*, 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. (1987). Duce, an oracle based approach to constructive induction. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pp. 287–292 Milan, Italy.

Muggleton, S., & Buntine, W. (1988). Machine invention of first-order predicates by inverting resolution. In *Proceedings of the Fifth International Conference on Machine Learning*, pp. 339–352 Ann Arbor, MI.

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.

Muslea, I., Minton, S., & Knoblock, C. (1998). Wrapper induction for semistructured, web-based information sources. In *Proceedings of the Conference on Automated Learning and Discovery.*

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 Linguistics*, pp. 40–47 Santa Cruz, CA.

Pazzani, M. J., Brunk, C. A., & Silverstein, G. (1992). An information-based approach to integrating empirical and explanation-based learning. In Muggleton, S. (Ed.), *Inductive Logic Programming*, pp. 373–394. Academic Press, New York.

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. (1993). *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo,CA.

Quinlan, J. R. (1994). The Minimum Description Length principle and categorical theories. In *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 233–241.

Quinlan, J. R. (1995). MDL and categorical theories (continued). In *Proceedings of the Twelfth International Conference on Machine Learning.*

Quinlan, J. R. (1996). Bagging, boosting, and C4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp. 725–730.

Quinlan, J. R., & Cameron-Jones, R. M. (1993). FOIL: A midterm report. In *Proceedings of the European Conference on Machine Learning*, pp. 3–20 Vienna.

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.

Ratnaparkhi, A. (1997). A linear observed time statistical parser based on maximum entropy models. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pp. 1–10.

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

Richmond, K., Smith, A., & Amitay, E. (1997). Detecting subject boundaries within text: A language independent stochastic approach. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pp. 47–54.

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.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, *14*, 465–471.

Sahami, M. (Ed.). (1998). *Proceedings of the AAAI98 Workshop on Learning for Text Categorization*, Madison, Wisconsin.

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.

Slattery, S., & Craven, M. (1998). Learning to exploit document relationships and structure: The case for relational learning on the web. In *Proceedings of the Conference on Automated Learning and Discovery.*

Smadja, F., McKeown, K. R., & Hatzivassiloglou, V. (1996). Translating collocations for bilingual lexicons: A statistical approach. *Computational Linguistics*, *22*(1), 1–38.

Soderland, S. (1998). Learning information extraction rules for semi-structured and free text. *Machine Learning*, *in press.*

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*, pp. 290–301. Springer, Berlin.

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.

Thompson, C. A., & Mooney, R. J. (1989). Semantic lexicon acquisition for learning natural language interfaces. In *Proceedings of the Sixth Workshop on Very Large Corpora.*

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.

Yang, Y., & Pederson, J. O. (1997). A comparitive study in feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 412–420.

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.

Zelle, J. M., Thompson, C., Califf, M. E., & Mooney, R. J. (1995). Inducing logic programs without explicit negative examples. In *Proceedings of the Fifth International Workshop on Inductive Logic Programming*, pp. 403–416 Leuven, Belgium.