LEARNING TEXT ANALYSIS RULES FOR DOMAIN-SPECIFIC NATURAL LANGUAGE PROCESSING

A Dissertation Presented

by

STEPHEN G. SODERLAND

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 1997

Department of Computer Science

LEARNING TEXT ANALYSIS RULES FOR DOMAIN-SPECIFIC NATURAL LANGUAGE PROCESSING

A Dissertation Presented

by

STEPHEN G. SODERLAND

Approved as to style and content by:

Wendy G. Lehnert, Chair

Paul E. Utgoff, Member

W. Bruce Croft, Member

Thomas Roeper, Member

David Stemple, Department Chair
Department of Computer Science

ACKNOWLEDGEMENTS

I would like to begin by thanking my advisor, Wendy Lehnert, for all of her guidance and encouragement. Under her direction the UMass Natural Language Processing Laboratory has been an ideal environment for creative work. With a large, complex text analysis system in place, it was natural to identify weaknesses of the system and focus effort on one component at a time. I got my feet wet with noun phrase analysis, did a Master's thesis on discourse analysis, and Doctoral research on CRYSTAL, a component that learns text analysis rules. Wendy gave me a free hand in developing CRYSTAL and put all of the resources of the lab at my disposal.

The person who first sparked my interest in computers and language deserves special mention. I took a series of seminars in 1970 from a young professor named Roger Schank at Stanford. Wendy was astonished to hear that I had known her advisor before she did. When I first applied to UMass and received copies of some research papers, I was impressed at how sensible their approach seemed to me: place the primary emphasis on meaning and let structure or linguistic theory take second place. Then I noticed references to Schank's work and realized why the approach at UMass seemed so right. Over the last five years that feeling of being in the right place has been confirmed again and again.

I would also like to thank the members of my committee, Bruce Croft, Tom Roeper, and especially Paul Utgoff, who guided me in understanding how my

work fits into the larger picture of machine learning research. I also want to thank Paul for his painstaking proofreading of my dissertation. My colleagues in the NLP lab, David Fisher and Joe McCarthy, were also a great help in proofreading and giving helpful comments on drafts of the dissertation.

A special, collaborative relationship has developed over the last two or three years with David Fisher. The CRYSTAL system would not have been written without his assistance. We bounced ideas back and forth on the design and the implementation, as well as bouncing puns, nonsense, and friendly insults back and forth. The implementation details are David's (I can hear him say, "Right, blame me, why don't you."). One of my regrets in graduating is that I can't take David with me.

Other members of our lab who have helped me over the years are Ellen Riloff, Claire Cardie, Joe McCarthy, Jon Aseltine, Fang-fang Feng and Jon Peterson. Ellen and Joe, in particular, became close friends and mentors. Priscilla Coe has kept our lab running smoothly by handling administrative matters, travel arrangements, and so forth. Expertise in applying CRYSTAL to a medical domain came from David Aronow, who helped define the information extraction task and supervised the team of nurses who annotated the training texts.

My thanks to the funding sources that made this work possible, the National Science Foundation, Library of Congress and Department of Commerce under cooperative agreement number EEC-9209623 and NRaD Contract Number N66001-94-D-6054. Any opinions, findings and conclusions or recommendations expressed in this material are mine and do not necessarily reflect those of the sponsors.

I would like to end by acknowledging support from familiy and friends. The curiosity and love of words that lies behind my research comes largely from my parents and brothers. We had an unabridged dictionary on a stand in the dining room, and a meal seldom went by without someone getting up to consult a word meaning or word origin. Thanks to Mark Rosenberg who convinced me that age forty was not really too old to go enter graduate school and take up the dream that I had since Professor Schank's seminar twenty years earlier. Special thanks to my wife, Noyuri, who has encouraged me and was willing to quit everything and move across country to Massachusetts. Her support has been a blessing to me.

ABSTRACT

LEARNING TEXT ANALYSIS RULES
FOR DOMAIN-SPECIFIC
NATURAL LANGUAGE PROCESSING

FEBRUARY 1997

STEPHEN G. SODERLAND

B.Sc., STANFORD UNIVERSITY

M.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Wendy G. Lehnert

An enormous amount of knowledge is needed to infer the meaning of unrestricted natural language. The problem can be reduced to a manageable size by restricting attention to a specific *domain*, which is a corpus of texts together with a predefined set of *concepts* that are of interest to that domain.

Two widely different domains are used to illustrate this domain-specific approach. One domain is a collection of Wall Street Journal articles in which the target concept is management succession events: identifying persons moving into corporate management positions or moving out. A second domain is a collection of hospital discharge summaries in which the target concepts are various classes of diagnosis or symptom.

The goal of an information extraction system is to identify references to the concept of interest for a particular domain. A key knowledge source for this purpose is a set of text analysis rules based on the vocabulary, semantic classes, and writing style peculiar to the domain.

This thesis presents CRYSTAL, an implemented system that automatically induces domain-specific text analysis rules from training examples. CRYSTAL learns rules that approach the performance of hand-coded rules, are robust in the face of noise and inadequate features, and require only a modest amount of training data.

CRYSTAL belongs to a class of machine learning algorithms called covering algorithms, and presents a novel control strategy with time and space complexities that are independent of the number of features. CRYSTAL navigates efficiently through an extremely large space of possible rules.

CRYSTAL also demonstrates that expressive rule representation is essential for high performance, robust text analysis rules. While simple rules are adequate to capture the most salient regularities in the training data, high performance can only be achieved when rules are expressive enough to reflect the subtlety and variability of unrestricted natural language.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

With the increasing amounts of on-line text available, the need is growing for automated text analysis systems that go beyond keywords to extract the conceptual content of the text. This requires a system that can reliably extract both the explicitly stated information and that which can be reasonably inferred.

The popular image in movies is of a person engaging in a philosophical discussion with a computer. The person tells the computer to read the collected works of Shakespeare to gain more insight into human behavior, which of course the computer does in about sixty seconds, understanding all the nuances.

Unfortunately, the amount of knowledge needed for in-depth understanding is overwhelming. The BORIS system [Lehnert *et al.* 1983], developed by Michael Dyer and other researchers at Yale in the 1980's gives an indication of just how much knowledge is needed. Three years of intensive knowledge engineering produced a system capable of impressive in-depth understanding of a two paragraph narrative, but unable to handle input other than those two paragraphs.

One approach to reducing the knowledge acquisition for text understanding to a manageable size has been that of *information extraction* (IE), in which

the task is restricted to identifying a predefined set of concepts in a specific *domain* and ignoring other information. A domain consists of a corpus of texts together with a clearly specified information need.

In recent years the Natural Language Processing Laboratory at the University of Massachusetts has developed IE systems for a wide variety of domains. One domain was a collection of news articles about Latin American terrorism, in which the target concepts to be extracted are terrorist events and the perpetrators, victims, physical targets, and weapons associate with those events. This domain was from the ARPA-sponsored Third and Fourth Message Understanding Conferences [MUC-3 1991, MUC-4 1992].

The MUC-5 conference [MUC-5 1993] used two domains. One was a Joint Ventures domain whose target concept involves companies forming joint business ventures. The other was a Microelectronics domain whose target concepts are the roles companies play in microchip fabrication technology.

The domains for MUC-3, MUC-4, and MUC-5 were all collections of news articles. Other domains have been collections of patients' medical records: a Hospital Discharge domain in which the target concepts are symptoms and diagnoses, and a Pediatric Asthma domain in which the target concept was acute asthma exacerbations. Another domain was a Goldfish Newsgroup domain that consisted of messages posted to a Usenet newsgroup for goldfish hobbyists. The target concept was identifying frequently asked questions about raising goldfish.

Some types of texts and some target concepts are more appropriate for an information extraction system than others. Newspaper articles are written in declarative sentences with a clear, professional writing style. This is much

more amenable to automatic processing than the often poorly written Usenet newsgroup postings.

Transcriptions of a physician's handwritten notes are often telegraphic and filled with idiosyncratic abbreviations and punctuation, making them unsuitable for IE technology. Typical entries from Pediatric Asthma notes are, "ASTHMA: STARTED SUNDAY-WELL CONTROLLED W/MEDS. CLEAR NOW" or "ASTHMA: MEDS D/C 6/2 DUE TO V,MILD WHEEZ THIS AM,NO DISTRESS PER MO". Dictated notes by a physician, such as those in Hospital Discharge texts, tend to use full sentences and are more accessible to an IE system.

The target concept must be clearly specified if an IE system is to be successful. Ideally, two independent human readers who are given the task of identifying references to the target concept in the same text should have nearly perfect agreement. If the target concept is so subtle or so ill-formed that human readers cannot reliably identify the concept, an automated system will not be able to do so.

A considerable amount of knowledge must be acquired for an automated text analysis system, knowledge that is highly specific to each domain. Different domains have different information needs, and also involve considerable differences in vocabulary and writing style.

This thesis presents CRYSTAL [Soderland *et al.* 1995], a system that automatically learns domain-specific rules for information extraction. Learning text analysis rules from examples is critical if information extraction is to be a feasible technology, since these rules are highly specific to a given domain and are difficult and time consuming to write by hand.

1.1 Domain-specific Text Analysis

Information extraction operates in the context of a clearly defined information need. To illustrate how an IE system works, let us consider the Management Succession domain, which was used in the MUC-6 performance evaluation [MUC-6 1995]. The task for this domain is to analyze news articles and identify persons moving into top corporate management positions and persons moving out of those positions. The only information considered relevant is the persons, positions, and corporations that are directly involved in a management succession event. Other persons, positions, and corporations are ignored as irrelevant to the domain.

The excerpt from a Wall Street Journal article in Figure 1.1 illustrates the type of information extracted from the Management Succession domain.

> Who's News: Topologix Inc.
>
> Donald E. Martella, formerly vice president, operations, was named president and chief executive officer of this maker of parallel processing subsystems. He succeeds Jack Harper, a company founder who was named chairman.
> ...

Figure 1.1 A text from the Management Succession domain

This text has three succession events: Donald Martella is moving into a position that Jack Harper is leaving; Martella is moving out of his old job as vice president; Harper is moving in as chairman. These succession events can be represented as three case frames, each case frame having up to four slots:

Person_In, *Person_Out*, *Position*, and *Organization*[1]. The output from this text is shown in Figure 1.2

```
Succession_Event:
    Person_In:      Donald E. Martella
    Person_Out:     Jack Harper
    Position:       president and chief executive officer
    Organization:   Topologix Inc.

Succession_Event:
    Person_Out:     Donald E. Martella
    Position:       vice president, operations
    Organization:   Topologix Inc.

Succession_Event:
    Person_In:      Jack Harper
    Position:       chairman
    Organization:   Topologix Inc.
```

Figure 1.2 Output from the sample text: three case frames

How can an information extraction system start from the raw text in Figure 1.1 and produce the desired output representation? This is done in several stages of processing, beginning with syntactic analysis that identifies syntactic constituents such as subject, verb phrase, direct object, and prepositional phrases. Each word is also assigned a semantic class.

At this point the IE system applies a set of domain-specific text analysis rules to identify references to relevant information. Rules that apply to the text in Figure 1.1 might look for patterns such as those in Figure 1.3.

The exact nature of these rules will vary from system to system, but all participants in the MUC evaluations included some form of rules that detect

[1]For the sake of clarity, I am using a somewhat simpler representation than the official MUC-6 output format.

1. "<Person> WAS NAMED <Position> OF <Organization>"
2. "<Person> SUCCEEDS <Person>"
3. "<Person> FORMERLY <Position>"
4. "<Person> WHO WAS NAMED <Position>"

Figure 1.3 Domain-specific patterns used for information extraction

relevant information based on local context. The rules used by CRYSTAL will be described in detail in Chapter 3.

Text analysis rules produce a fragmentary view of the text. For example, a rule based on the first pattern "<Person> WAS NAMED <Position> OF <Organization>" would identify Donald E. Martella as *Person_In* with a *Position* of president and chief executive officer, but would leave the *Person_Out* blank and find only a generic reference for the *Organization*, as shown below.

Succession_Event:
Person_In: Donald E. Martella, formerly vice president, operations
Position: president and chief executive officer
Organization: this maker of parallel processing subsystems

Figure 1.4 Information extracted using pattern 1

A rule that looks for the pattern "<Person> SUCCEEDS <Person>" would identify "He" as the *Person_In* and Jack Harper as *Person_Out*, but would not be able to determine the *Position* or *Organization*.

Succession_Event:
Person_In: He
Person_Out: Jack Harper, a company founder

Figure 1.5 Information extracted using pattern 2

An IE system needs to consolidate the output of the text analysis rules in a later step known as *discourse processing*. This trims extraneous terms from case frame slots, handles coreference resolution of pronouns and generic references, merges related case frames, and infers values for empty slots.

This thesis will concentrate on the rules that extract information based on local context, keeping in mind that this is only one of many components in a full IE system.

1.2 The Need for Trainable Systems

Text analysis rules developed for one domain cannot, in general, be transferred to a new domain. Rules to extract Management Succession events are of no use in a medical domain. CRYSTAL was also tested on a Hospital Discharge domain in which the relevant information is symptoms and diagnoses in patient hospital records. Rules for this domain are based on patterns such as the following, which would apply to the input "Chest x-ray revealed a new right pleural effusion".

"<Diagnostic Procedure> REVEALED <Finding>"

Figure 1.6 A pattern used in extraction from a medical domain

These rules also depend on appropriate semantic class assignment of individual words. The IE system needs a semantic lexicon that assigns the class <Diagnostic Procedure> to "x-ray" and the class <Finding> to "pleural effusion". Even given such semantic class assignment, creating a sufficient set of reliable text analysis rules is a difficult and time-consuming task.

Building rules by hand requires both system expertise and domain expertise. This is particularly clear for a highly technical domain such as the Hospital Discharge domain. A computational linguist may understand the text analysis software, but lack the medical knowledge needed to create reliable text analysis rules. A medical professional, on the other hand, will understand the terminology and medical reasoning, but have no idea how to translate that into rules for a computer system.

A reasonable approach, and the paradigm adopted by the MUC evaluations, is corpus-based system development. Domain experts[2] take a representative set of texts and annotate them by hand to create an answer key for each text. This corpus of annotated texts defines the relevant information by example, and is used by system developers to guide development of text analysis rules.

If some of the annotated information is missed by the rules, this indicates the need for new rules or for broadening of existing rules. If information is extracted that was not marked as relevant by a domain expert, this indicates that a rule is overly general and is creating errors.

CRYSTAL automates this method of corpus-based rule generation. A set of hand-annotated texts serves as an implicit definition of relevant information. During induction of a rule base, CRYSTAL looks for an instance of relevant information not covered by existing rules. This instance becomes a seed for the creation of a new rule. CRYSTAL tests each rule that it proposes against the entire training corpus to ensure that the rule has not been generalized too far.

[2]Domain experts for a medical domain would be physicians or nurses

The goal of CRYSTAL is a turnkey system that can be easily adapted to the information needs of an end user. The user defines an information extraction task by annotating a set of training texts. This does not require expertise in linguistics or computer science. CRYSTAL then induces a set of text analysis rules automatically.

1.3 Claims

This thesis centers around CRYSTAL, a system that learns text analysis rules from training examples. CRYSTAL makes contributions to natural language processing and to machine learning.

Claim 1:

> CRYSTAL demonstrates that high quality text analysis rules can be learned from examples, rules which:
>
> a. approach the performance of hand-coded rules
> b. are robust in the face of noise and inadequate features
> c. require modest training size

Empirical results from Chapter 5 show performance nearly as high as that of hand-coded rules, depending on the difficulty of the concept being learned and the adequacy of the training data. Experiments were conducted on two versions of the Hospital Discharge data with differing amounts of noise. Semantic tagging of words in CRYSTAL's input for one data set are based on a generic on-line thesaurus. The other has semantic tagging customized for the information extraction task. Other experiments show CRYSTAL's performance with various amounts of training data.

Claim 2:

> CRYSTAL presents a covering algorithm control strategy that navigates efficiently in extremely large feature spaces:
>
> a. time and space complexity independent of the feature size
> b. a greedy approach that can be improved very little by using more extensive search

CRYSTAL's machine learning algorithm belongs to the family of covering algorithms, which is described in Chapter 8. CRYSTAL is unique in its ability to handle the extremely large feature sets that can arise in natural language processing problems. For example, a feature might represent any of thousands of words occurring in any of several syntactic roles. Experiments in Chapter 5 also show that increasing the amount of search effort does little to improve CRYSTAL's performance, and in some cases degrades performance.

Claim 3:

> CRYSTAL demonstrates that expressive representation is essential for high performance, robust text analysis rules.

There is an inherent tension between the need for expressive representation and the increased difficulty for a machine learning algorithm if the representation is overly complex. Experiments in Chapter 6 show that performance suffers when various aspects of CRYSTAL's representation language are disabled.

The richness and flexibility of CRYSTAL's rule representation proved to be important for performance in nearly every case. This is particularly true when faced with noisy data and sparse training.

1.4 Organization of the Dissertation

The remainder of the dissertation is organized as follows. Chapter 2 presents the two contrasting domains that will be used to demonstrate CRYSTAL. These are the Management Succession domain and the Hospital Discharge domain, both of which have been briefly introduced in this chapter.

The CRYSTAL algorithm is introduced in Chapters 3 and 4. Chapter 3 describes the text analysis rules that CRYSTAL learns, and Chapter 4 presents the induction algorithm. This is followed by a series of experiments that test various aspects of CRYSTAL empirically.

Chapter 5 shows performance in each of the two test domains and presents experiments that assess how close CRYSTAL comes to finding optimal rules, including comparisons with results from hand-coded rules. Parameter settings allow a user to optimize for either recall or for precision[3]. Another experiment shows that CRYSTAL's efficient algorithm does not benefit from increasing the amount of search effort.

Chapter 6 explores the impact of CRYSTAL's expressive representation by enhancing and by restricting CRYSTAL's rule representation.

Related work in natural language processing and in machine learning is presented in Chapters 7 and 8, respectively. Chapter 9 revisits the main claims of the thesis, presents conclusions, and points to future work.

Five appendices are included. Appendices A and B have tables of empirical results for each of the two domains used to test CRYSTAL. Appendices C and

[3]Recall is the percentage of possible information that the system finds. Precision is the percentage of information reported that is correct. These metrics are defined more precisely in Chapter 5.

D have the semantic hierarchy for each domain. Appendix E list text analysis rules derived by CRYSTAL for a Management Succession concept.

CHAPTER 2

TEST DOMAINS

CRYSTAL has been applied successfully to several domains. I have chosen two domains with sharply contrasting characteristics as a test bed for the experiments presented in this dissertation. The Management Succession domain involves business-related news articles, while the Hospital Discharge domain involves medical records with a specialized medical vocabulary.

2.1 The Management Succession Domain

The Management Succession domain was briefly introduced in Chapter 1. This domain is a corpus of Wall Street Journal texts, in which the information to be extracted concerns persons moving into and out of top corporate management positions.

It is important to be clear about what is *not* considered relevant in this domain. Only top management positions in corporations are relevant: not governmental appointments or union positions. Being a member of the board of directors is not relevant, although chairmanship of a company is relevant.

Management Succession output is represented as case frames with four slots: *Person_In*, *Person_Out*, *Position*, and *Organization*. One example of Management Succession output has already been presented in Figure 1.2. The

Input text:

> Staar Surgical Co.'s board said that it has removed
> Thomas R. Waggoner as president and chief executive
> officer and that John R. Wolf, formerly executive
> vice president, sales and marketing, has been named
> president and chief executive officer.
> ...
> The Staar board also said that John R. Ford resigned
> as a director, and that Mr. Wolf was named a member
> of the board.

Output representation:

Succession_Event:

Person_In:	John R. Wolf
Person_Out:	Thomas R. Waggoner
Position:	president and chief executive officer
Organization:	Staar Surgical Co.

Succession_Event:

Person_Out:	John R. Wolf
Position:	executive vice president, sales and marketing
Organization:	Staar Surgical Co.

Figure 2.1 Output from a Management Succession text (being named to or leaving the board of directors is ignored as irrelevant to the domain)

text in Figure 2.1 shows another text, which has a mixture of relevant and irrelevant appointments and removals from office.

Sometimes judging whether an event is relevant cannot be done solely on the basis of local context. The sentence "He succeeded George Adams, who resigned in July" may or may not be a management succession event. If Mr. Adams resigned as a corporate officer, then it is relevant. If he was head of a government agency, editor of a newspaper, or member of a board of directors,

then it is not relevant. Text analysis rules that are based on local context cannot make such distinctions and must rely on later discourse processing to filter out incorrect extractions.

CRYSTAL requires a semantic hierarchy for each domain that shows which semantic class is a subclass of another class. The Management Succession domain needs semantic classes such as <Person Name> ("Mr. Smith") and <Generic Person> ("he"), which are subclasses of <Person>. The classes <Organization Name> ("IBM") and <Generic Organization> (the "company") are likewise subclasses of <Organization>. <Corporate Post> ("chief executive officer") and <Generic Position> (the "post") are subclasses of <Position>. Appendix C shows the entire hierarchy of 55 semantic classes.

The corpus of Management Succession texts includes the 200 texts provided by the MUC-6 performance evaluation. I felt that 200 texts might provide insufficient training examples, so I expanded the corpus with additional texts from the Wall Street Journal. The texts include a mix of actual management succession events as well as "near misses" involving irrelevant positions such as director or editor and irrelevant organizations such as government agencies.

The training for this domain consists of 599 annotated texts with the statistics shown in Figure 2.1[1].

2.2 The Hospital Discharge Domain

Hospital discharge summaries are dictated by a physician at the conclusion of a patient's hospitalization. The writing style and vocabulary include terms

[1] A sentence may result in multiple "training instances" depending on how it is syntactically analyzed. For example a sentence with multiple independent clauses becomes multiple instances.

Texts:	599	Person_In:	678
Sentences:	10,053	Person_Out:	714
Instances:	16,325	Position:	1,025
		Organization:	833

Table 2.1 Statistics on number of sentences and instances in the Management Succession domain

and usages peculiar to medical notes and contain a mixture of full sentences and sentence fragments.

The task in this domain is to identify all references to symptoms and to diagnoses. These are further broken down into four categories:

Symptom, Present

Symptom, Absent

Diagnosis, Confirmed

Diagnosis, Ruled_Out

Symptom includes both clinical findings and statements by a patient about his or her condition. Phrases that indicate an abnormal condition are instances of *Symptom,Present*, while normal or unremarkable findings are instances of *Symptom,Absent*.

Diagnosis means a conclusion drawn by a physician. If the text explicitly states that the patient does not have a disease or allergy, this is classified as *Diagnosis,Ruled_Out*. Other diagnoses are considered *Diagnosis,Confirmed*, whether the diagnosis was made during the current hospitalization or was previously made. Medical conditions of family members or conditions that are only suspected are considered irrelevant and not extracted.

Figure 2.2 shows an excerpt of a Hospital Discharge text. The output has nine separate case frames, one for each extracted fact. Unlike the Management Succession domain, in which the output representation has multi-slot case frames, the Hospital Discharge domain has only single-slot case frames.

Input text:

> HISTORY OF PRESENT ILLNESS: ... He also has a medical history significant for cirrhosis and on a recent screening chest X-Ray, was found to have new right sided lung nodules. ... He also is complaining of night sweats but denies any chest pain, hemoptysis, or shortness of breath. ...
> ALLERGIES: He has no known drug allergies.
> PHYSICAL EXAMINATION: ... LUNGS: Initially had diffuse rhonchi and all fields cleared after coughing.

Output representation:

> Diagnosis, Confirmed: cirrhosis
>
> Symptom, Present: new right sided lung nodules
>
> Symptom, Present: night sweats
>
> Symptom, Absent: chest pain
>
> Symptom, Absent: hemoptysis
>
> Symptom, Absent: shortness of breath
>
> Diagnosis, Ruled_Out: no known drug allergies
>
> Symptom, Present: diffuse rhonchi
>
> Symptom, Absent: all fields cleared

Figure 2.2 A Hospital Discharge text and a list of extracted facts

This is a major difference between the two domains. Information in the Management Succession domain centers around events. Individual facts such as persons, positions, and organizations are relevant only because of a relationship between the facts. Extracted facts in the Hospital Discharge domain, however, stand in isolation from each other. A phrase such as "lung nodules" will always be a *Symptom,Present* when it occurs in an affirmative context (e.g. "was found to have lung nodules") and a *Symptom,Absent* if negated (e.g. "exam revealed no lung nodules").

The Hospital Discharge domain requires a semantic hierarchy appropriate to medical texts. CRYSTAL uses a hierarchy of 133 semantic classes adapted from the Unified Medical Language Systems (UMLS) on-line medical thesaurus, developed by the National Library of Medicine [Lindberg *et al.* 1993]. In this hierarchy a <Finding> has two subclasses, <Sign or Symptom> and <Laboratory or Test Result>. A <Finding> is a <Conceptual Entity>. The semantic class <Disease or Syndrome> is a <Pathologic Function>, which is a descendant of the class <Event>. Appendix D shows the entire hierarchy.

The Hospital Discharge corpus consisted of 502 texts with the following characteristics. This domain is denser in information content than Management Succession, with twice as many slot fills per sentence.

Texts:	502	Symptom, Present:	3,915
Sentences:	15,250	Symptom, Absent:	4,309
Instances:	17,500	Diagnosis, Confirmed:	2,100
		Diagnosis, Ruled_Out:	446

Table 2.2 Statistics on number of sentences and instances in the Hospital Discharge domain

2.3 Annotating the Training Corpora

CRYSTAL is a supervised learning algorithm, and as such needs training instances that have been annotated by a human expert. CRYSTAL is given a training set of texts in which every instance of the concept being learned (e.g. management succession event) has been explicitly marked in the text. Any phrases not marked as positive instances of the target concept are considered to be negative instances.

CRYSTAL's job is to induce a set of general rules from this training that will allow it to identify the target concept in previously unseen texts. In effect, CRYSTAL is learning to imitate the human experts who annotated the training material.

Training texts for these experiments were annotated by marking relevant phrases, to label that phrase's role in a target concept. A point-and-click interface was developed to facilitate the process by colleagues in the University of Massachusetts Natural Language Processing Laboratory[2].

The sentence in Figure 2.3 has been annotated to show that "night sweats" is a *Symptom,Present* (<SP>) and that "chest pain", "hemoptysis", and "shortness of breath" are *Symptom,Absent* (<SA>).

The annotation for the Hospital Discharge domain was done by a team of three nurses under the supervision of the physician who helped to define the information extraction task. They were able to mark an average of five texts per hour.

[2]The text marking interface was developed David Fisher and Fang-fang Feng. It allows a user to specify a set of short abbreviations for concept slot names. The user then indicates a word or phrase to be labeled with one or more of those abbreviations. The labeling conventions resemble those used for hypertext links. <SP> marks the start of a *Symptom,Present* and </SP> marks the end of the labeled phrase.

He also is complaining of <SP> night sweats </SP> but denies
any <SA> chest pain </SA>, <SA> hemoptysis </SA>, or
<SA> shortness of breath </SA>.

Figure 2.3 Training annotations in a Hospital Discharge text

In domains with output represented as multi-slot case frames, annotation must also indicate which phrases participate in the same case frame. In the Management Succession domain, phrases that play a role in succession event 1 are labeled with "SE=1", and so forth. Figure 2.4 shows a sentence with two succession events.

<PI SE=1> He </PI> succeeds <PO SE=1> <PI SE=2> Jack Harper
</PI> </PO>, a <SO SE=2> company </SO> founder who was
named <SP SE=2> chairman </SP>.

Figure 2.4 Training annotations for Management Succession

In this sentence, succession event 1 (SE=1) has two slots filled: "he" is *Person_In* and "Jack Harper" is *Person_Out*. Mr. Harper plays the role of *Person_In* in succession event 2 (SE=2), which also has an *Organization* (SO) and a *Position* (SP).

Note that generic references and pronouns are marked as valid phrases to extract. This is done under the assumption that later discourse processing will resolve these to an actual person or company name. Another decision that I made in creating training for Management Succession was to mark only information that could be inferred from the context of the current sentence. Once I had settled on annotation guidelines, text marking went quickly, at an average of 20 texts per hour.

Training annotations are added directly to the texts and make no assumptions about what syntactic analysis and semantic tagging will later be done by the information extraction system. The annotation is also neutral about the representation language of the text analysis rules. CRYSTAL's rule representation is presented in the following chapter.

CHAPTER 3

CRYSTAL'S TEXT EXTRACTION RULES

CRYSTAL learns rules to extract concepts of interest to a particular domain (e.g. succession events) based on local linguistic context. These rules use a combination of syntactic, semantic, and lexical evidence to identify references to the target concept.

How should local linguistic context be represented in the instances presented as input to CRYSTAL and in CRYSTAL's text analysis rules? The representation must be expressive enough to capture the richness of natural language, yet not so complex that a machine learning algorithm has trouble finding regularities in the training.

The rules must have access to some level of syntactic knowledge, at least distinguishing major syntactic constituents such as subject, verb, and direct object. Otherwise CRYSTAL could not tell who is the *Person_In* and who is the *Person_Out* in "Mr. A succeeds Mr. B as chairman" and "Mr. B succeeds Mr. A as chairman". The rules will also need to distinguish affirmative from negative phrases and distinguish active from passive verbs.

Expressing rules in terms of semantic classes allows compact rules with greater generality than rules using only exact words. If semantic classes have been assigned to words in the input, a rule can, for example, require the class <Corporate Post> to be found in a prepositional phrase. This would apply to

instances where someone has been named "as chairman", "as chief executive officer", "as vice president", and so forth. Semantic classes allow a single rule to cover a long list of exact words.

The semantic class assignment must be appropriate to the domain. Rules for the Management Succession domain can be expressed in terms of semantic classes such as <Person Name>, <Organization Name>, and <Corporate Post>. The Hospital Discharge domain needs classes such as <Disease or Syndrome>, <Body Part>, and <Finding>.

Semantic classes alone may not be sufficient in some cases, however. In the Management Succession domain, the word "former" is a good clue that someone is a *Person_Out* ("the former chairman"), while "new" is evidence for *Person_In* ("the new chairman"). Using exact words as well as semantic classes is especially important when the semantic class assignment has not been fine-tuned to make the distinctions needed for the information extraction task.

Another consideration that applies to both semantic classes and exact words is the distinction between head terms and modifying terms. The meaning of a word or class may depend on whether it is found as a head or modifier. For example, the term "cancer" in a hospital discharge report is usually evidence of the concept *Diagnosis*, but not when it is used as a modifier ("cancer studies" or "cancer treatment").

The following section shows how CRYSTAL incorporates these various types of linguistic evidence in the representation of instances and of rules.

3.1 Concept Definitions

CRYSTAL's text analysis rules, called *concept definitions*, do not operate directly on the raw text. The text is first processed by a sentence analyzer to produce *instances* for CRYSTAL. These instances have been segmented into syntactic constituents such as subject, verb, object, and prepositional phrases. In addition, each word has been tagged with a semantic class.

The concept definitions apply semantic and lexical *constraints* to syntactic constituents of the instance. These constraints are conditions that must be satisfied by the instance in order for the rule to fire. If all constraints are satisfied, CRYSTAL creates a case frame representing information extracted from that instance.

Some examples of constraints are a lexical constraint that requires a verb phrase to include the words "succeeded", a semantic constraint that requires a direct object to include the semantic class <Person Name>, and a third constraint that requires a prepositional phrase to include the class <Corporate Post>.

These three constraints are all satisfied by a sentence such as "He succeeded George Henshaw as chairman and chief executive officer." A concept definition for the Management Succession domain might use these constraints to identify instances that have a *Person_Out* in the direct object and *Position* in the prepositional phrase.

CRYSTAL is independent of the syntactic analysis and the semantic class assignment. It uses whatever syntactic labels and semantic classes are found in the training instances. Many sentence analyzers transform a sentence into a

deeply nested parse tree. The present implementation of CRYSTAL, however, requires that the instances be presented as a flat list of syntactic constituents with no nested structure.

Training instances for these experiments were created by the BADGER sentence analyzer of the University of Massachusetts [Fisher *et al.* 1995]. The Hospital Discharge domain used a version of BADGER that segments each simple clause into constituents such as SUBJ, VERB, OBJ, ADV, and PP (prepositional phrase). The version used for the Management Succession domain uses those syntactic constituents and also has constituents labeled REL-SUBJ, REL-VERB, REL-OBJ, and REL-PP for relative clauses attached to the subject, verb, object, and prepositional phrase, respectively.

In addition to training instances, CRYSTAL is given a semantic hierarchy for the domain and a data file listing the concepts for the domain. The semantic hierarchy allows rules with class constraints to cover subclasses, for example a constraint requiring the class <Person> covers the subclasses <Person Name> and <Generic Person> in Management Succession instances.

Consider the following sentence, "Paul Herold, who was formerly a senior vice president, was recently named chairman of this major pharmaceuticals concern." Two concept definitions that apply to this sentence are shown in Figure 3.1. The first identifies Mr. Herold as a *Person_Out* of a *Position* found in a relative clause attached to the subject. The second concept definition identifies him as a *Person_In* to a *Position* found in the direct object.

These concept definitions illustrate the constraints CRYSTAL may apply to syntactic constituents of an instance. The terms constraint is an unordered list of words that must be included in the syntactic constituent. The classes

```
Concept type: Succession Event
Constraints:
   SUBJ::
      Classes include:  <Person Name>
      Extract:          Person_Out
   REL-SUBJ::
      Terms include:    FORMERLY
      Classes include:  <Corporate Post>
      Extract:          Position


Concept type: Succession Event
Constraints:
   SUBJ::
      Classes include:  <Person Name>
      Extract:          Person_In
   VERB::
      Root:             NAME
      Mode:             passive
   OBJ::
      Classes include:  <Corporate Post>
      Extract:          Position
```

Figure 3.1 Two concept definitions that apply to "Paul Herold, who was formerly a senior vice president, was recently named chairman of this major pharmaceuticals concern."

constraint is an unordered list of semantic classes that must be present, either directly or through an IS-A relationship.

Lexical and semantic constraints may also make a distinction between head terms or classes and modifier terms or classes. Terms found as the last term of the phrase, or just before punctuation, before a preposition, or before an adverb are considered to be head terms. All others are considered modifiers.

The mode constraint is used for "affirmative" or "negative", "active" or "passive". CRYSTAL will accept whatever modes it finds in its input in-

stances, as assigned by the sentence analyzer. The root constraint uses any morphological information supplied by the sentence analyzer. Concept definitions with constraints on a prepositional phrase may also have a constraint that requires a particular preposition.

Here is a list of constraints used by CRYSTAL.

Constraints on syntactic constituents:
- Terms
- Head terms
- Modifier terms
- Classes
- Head classes
- Modifier classes
- Root
- Preposition
- Mode (affirmative/negative, active/passive)

Figure 3.2 Constraints in a concept definition

If all constraints for each syntactic constituent are satisfied, CRYSTAL creates a case frame for the target concept. The concept definition specifies the output concept and associates each case frame slot with a syntactic constituent. For example, the first concept definition in Figure 3.1 fills the *Person_Out* slot with the subject and the *Position* slot with the REL-SUBJ. Later processing will generally be needed to trim extraneous words from the extracted phrases.

3.2 A Few Sample Concept Definitions

Figure 3.3 shows a sentence that has been syntactically segmented and semantically tagged to produce an instance for CRYSTAL. Note that the phrase

Input Sentence:
He succeeds Jack Harper, a company founder who was named chairman.

CRYSTAL Instance:
SUBJ:
Terms: HE
Classes: <Generic Person>
Mode: affirmative
VERB:
Terms: SUCCEEDS
Root: SUCCEED
Classes: <Root Class>
Mode: active, affirmative
OBJ:
Terms: JACK_HARPER %COMMA% A COMPANY FOUNDER
Classes: <Person Name>, <Generic Organization>, <Generic Person>
Mode: affirmative
REL-OBJ:
Terms: WHO WAS NAMED CHAIRMAN %PERIOD%
Classes: <Past>, <Event>, <Corporate Post>
Mode: affirmative

Figure 3.3 Syntactic analysis and semantic tagging of Management Succession input

"who was named chairman" is included in the instance as a REL-OBJ (relative clause attached to the direct object).

The instance also has semantic classes assigned to each word. The words "he", "company", "founder", "was", "named", and "chairman" are all found in the semantic lexicon for this domain, although "succeeds" was not, due to an oversight. The semantic lexicon and semantic hierarchy were created specifically for the Management Succession domain. Words not found in the semantic lexicon are given the class <Root Class>.

In the Management Succession domain, BADGER augments its semantic lexicon with a proper name recognizer that identifies persons, organizations, and locations. This module has assigned the semantic class <Person Name> to "Jack Harper".

Figure 3.4 shows a concept definition that applies to the instance in Figure 3.3. This definition tests that the subject contains a word with semantic class <Person>, that the verb root is "succeed", and that the direct object contain the class <Person Name>. If all these constraints are met CRYSTAL creates a *Succession Event* case frame with the subject of the instance in the *Person_In* slot and the direct object in the *Person_Out* slot.

```
Concept type: Succession Event
Constraints:
   SUBJ::
      Classes include:   <Person>
      Extract:           Person_In
   VERB::
      Root:              SUCCEED
      Mode:              active
   OBJ::
      Classes include:   <Person Name>
      Extract:           Person_Out
```

Figure 3.4 A concept definition that applies to "He succeeds Jack Harper, a company founder ..."

CRYSTAL consults a domain-specific semantic hierarchy to test the semantic constraints. The semantic constraint on the subject is met by "he" since the class <Generic Person> is a subclass of <Person>. All the other constraints are also satisfied by this instance, so CRYSTAL creates a case

frame with *Person_In* filled by "He" and *Person_Out* filled by "Jack Harper, a company founder".

Figure 3.5 shows a second concept definition that applies to the same instance. This definition looks for *Person_In* in the direct object and *Position* in a relative clause attached to the direct object. Semantic constraints require a <Person Name> in the direct object and a <Corporate Post> in the relative clause. The relative clause must also include the terms "who" and "named".

```
Concept type: Succession Event
Constraints:
   OBJ::
      Classes include:   <Person Name>
      Extract:           Person_In
   REL-OBJ::
      Terms include:     WHO NAMED
      Classes include:   <Corporate Post>
      Extract:           Position
```

Figure 3.5 A concept definition that applies to "... Jack Harper, a company founder who was named chairman"

After applying this concept definition to the instance in Figure 3.3, CRYSTAL creates a case frame with the *Person_In* slot filled by the phrase "Jack Harper, a company founder". The *Position* slot is filled by the phrase "who was named chairman". Each of these phrases includes the desired information, but also has extraneous words that must be trimmed away by later processing in a full information extraction system.

3.3 Finding the Right Level of Generalization

These concept definitions operate properly on the sample instance, but so would a large range of possible concept definitions. The definition in Figure 3.4 has a class constraint requiring <Person> in the subject. A more restrictive constraint would work just as well on this particular instance. The constraint on the subject could require the class <Generic Person> and further require that there be no modifier term. Figure 3.6 shows a concept definition with these constraints.

```
Concept type: Succession Event
Constraints:
   SUBJ::
      Modifier terms include:   <null>
      Head classes include:     <Generic Person>
      Extract:                  Person_In
   VERB::
      Root:                     SUCCEED
      Mode:                     active
   OBJ::
      Classes include:          <Root Class>
      Extract:                  Person_Out
```

Figure 3.6 An alternate concept definition for "He succeeds Jack Harper, a company founder"

The definition in Figure 3.6 also drops all constraints on the direct object. Constraining the direct object to have a <Person> is unnecessary, since a sentence beginning "He succeeds" can have nothing other than a person in the direct object in this domain.

I am not implying that Figure 3.6 is a better concept definition than that in Figure 3.4 or vice versa. This example is merely intended to point out

that a variety of concept definitions can apply to a given instance. Some of the possible definitions may seem foolish, such as one with a term constraint requiring a "%comma%" in the direct object. The total number of possible concept definitions that may be derived from this instance is astonishingly large, as will be shown in the following chapter.

The challenge for CRYSTAL is to derive concept definitions from training examples that are neither too restrictive nor too general. An overly restrictive definition will be unlikely to apply to new instances, for example constraining the direct object to include "Jack Harper". On the other hand, an overly general definition will make extraction errors and apply to instances that do not contain the target concept. The next chapter presents CRYSTAL's strategy for finding the appropriate level of generalization.

CHAPTER 4

CRYSTAL'S INDUCTION ALGORITHM

The goal of CRYSTAL is to find a set of concept definitions that are generalized enough to have good coverage on previously unseen texts, yet constrained tightly enough to operate reliably. CRYSTAL's approach is to begin with highly specific concept definitions and gradually relax the constraints. Each proposed generalization is tested for extraction errors on the training set, which has been hand-tagged with the desired phrases to be extracted. Generalization continues until further relaxation would lead to a definition that exceeds a user-specified error tolerance.

CRYSTAL begins by selecting a positive instance of the target concept as a seed. CRYSTAL then takes the most specific concept definition that covers this instance and generalizes it. Each proposed generalization is tested on the training set to ensure that the proportion of negative instances does not exceed a user-specified error tolerance.

The most general definition within error tolerance is added to the rule base and another seed is selected from positive instances not yet covered by the rule base. This is repeated until all positive instances have been covered or have been selected as seed instances. This methodology in machine learning is called a *covering algorithm* (see Chapter 8).

4.1 Deriving Initial Concept Definitions

The first step of CRYSTAL is to create a set of initial concept definitions, one from each positive instance of the concept being learned. An initial definition is the most restrictive one that covers the instance: every term and every semantic class is required to be identical to the motivating instance. CRYSTAL includes all words and classes, since it does not know in advance which of these features are essential to the concept and which will later be dropped during generalization.

In a multi-slot concept such as *Succession Event*, some instances will contain only a subset of the case frame slots. In the Management Succession domain, only 9% of the sentences referring to a succession event contain references to all four slots. If the company name has been mentioned in a previous sentence, only the *Person_In* and *Position* might be mentioned, for example.

The most useful approach for information extraction is to have CRYSTAL learn each subset of slots as a separate concept to be learned[1]. Any instance that contains that subset of slots is considered a positive instance of the target concept. For example, one target concept is succession events that have a *Person_In* and a *Person_Out* slot. Instances that contain at least these two slots are positive. All other instances are considered negative instances for this concept, even though they may be positive for other concepts.

Figure 4.1 shows an initial concept definition from the input "He succeeds Jack Harper, a company founder who was named chairman." This sentence

[1]This results in $2^k - 1$ combinations of a k-slot concept. A user can specify which of these combinations are to be learned. For the experiments with the Management Succession domain, I had CRYSTAL learn all 15 combinations

has been hand-tagged with a succession event containing "He" as *Person_In* and "Jack Harper" as *Person_Out*[2].

This initial concept definition will correctly identify that the subject of the sentence contains a *Person_In* and the direct object contains a *Person_Out*, but is so restrictive that it will not be satisfied by a sentence in any other text. In machine learning terminology, it is the maximally specific concept description that covers this instance. CRYSTAL must learn which of these constraints to relax.

Term constraints can be relaxed by dropping terms. The direct object constraint requiring "Jack_Harper %comma% a company founder" has five terms (Jack_Harper is treated as a single term, as is %comma%). There are 32 possible ways to relax this constraint by dropping a subset of the terms ($2^5 = 32$). There are also four possible relaxations of the two-word head terms constraint and eight for the three-word modifier terms constraint ($2^2 = 4$, $2^3 = 8$). This is equivalent to generalizing a formula in the predicate calculus by dropping a conjunctive term.

Class constraints may be relaxed by moving up in the semantic hierarchy. For example <Generic Person> or <Person Name> may be relaxed to <Person>, then to <Entity>, then to <Root Class>, which is equivalent to no constraint. There are 16 distinct relaxations possible for the classes con-

[2]See Figure 2.4 for the tagged sentence and Figure 3.3 for the syntactic analysis that forms the basis for a training instance.

Concept type: Succession Event
Constraints:

SUBJ::	
Terms include:	HE
Head terms include:	HE
Modifier terms include:	<null>
Classes include:	<Generic Person>
Head classes include:	<Generic Person>
Mode:	affirmative
Extract:	Person_In
VERB::	
Terms include:	SUCCEEDS
Head terms include:	SUCCEEDS
Modifier terms include:	<null>
Root:	SUCCEED
Mode:	active, affirmative
OBJ::	
Terms include:	JACK_HARPER %COMMA% A COMPANY FOUNDER
Head terms include:	JACK_HARPER FOUNDER
Modifier terms include:	%COMMA% A COMPANY
Classes include:	<Person Name>, <Generic Organization>, <Generic Person>
Head classes include:	<Person Name>, <Generic Person>
Modifier classes include:	<Generic Organization>
Mode:	affirmative
Extract:	Person_Out
REL-OBJ::	
Terms include:	WHO WAS NAMED CHAIRMAN %PERIOD%
Head terms include:	CHAIRMAN
Modifier terms include:	WHO WAS NAMED %PERIOD%
Classes include:	<Past>, <Event>, <Corporate Post>
Head classes include:	<Corporate Post>
Modifier classes include:	<Past>, <Event>
Mode:	affirmative

Figure 4.1 The initial concept definition for "He succeeds Jack Harper, a company founder ..."

straint on the direct object[3], 6 for the head classes constraint, and 4 for the modifier classes constraint.

With over one thousand possible ways[4] to relax constraints on the direct object, over one thousand for the REL-OBJ, and over one hundred each for the subject and the verb, there are more than one billion ways to generalize the initial concept definition in Figure 4.1.

CRYSTAL must search among this enormous space of possible concept definitions for an acceptable level of generalization. CRYSTAL needs a search control strategy that allows it to drop quickly those constraints that are merely accidental features of the motivating instance and to retain those constraints that are essential to the target concept.

4.2 Generalizing the Initial Definitions

CRYSTAL gradually relaxes constraints on the initial concept definition, which typically covers only a single positive instance. Each generalization step relaxes constraints enough to increase the number of positive training instances covered. Each proposed generalization is then tested on the entire training set to ensure that it does not cover an excessive proportion of negative training instances.

[3]Let GP=<Generic Person>, PN=<Person Name>, P=<Person>, GO=<Generic Organization>, O=<Organization>, E=<Entity>, R=<Root Class>. There are 16 distinct relaxations of the classes constraint: (GP,PN,GO), (GP,PN,O), (GP,PN,E), (GP,P,GO), (GP,P,O), (GP,P,E), (P,PN,GO), (P,PN,O), (P,PN,E), (P,GO), (P,O), (P,E), (E,GO), (E,O), (E), (R).

[4]There are 32 possible relaxations for the terms constraint, 4 for head terms, 8 for modifier terms, 16 for the classes constraint, 6 for head classes, 4 for modifier classes, and 2 for the mode constraint. If these are relaxed independently there are 32 x 4 x 8 x 16 x 6 x 4 x 2 = 786,432 combinations. Some of these are functionally equivalent, but there are several thousand distinct relaxations for constraints on the direct object.

The key insight of CRYSTAL is to guide this relaxation by finding the most similar initial concept definition. CRYSTAL creates a proposed generalization by dropping constraints that are not shared by the similar definition. This is equivalent to relaxing constraints just enough to cover the most similar positive instance, since each initial concept definition corresponds to a positive training instance.

Unifying the term constraint "Harold_Archer %comma% former chairman of Atlas" with "Mr. Green %comma% chairman since 1982" results in dropping all terms from the constraint but "%comma%" and "chairman". In the case of class constraints, unification means moving up in a semantic hierarchy to a common ancestor. For example, unifying <Person Name> with <Generic Person> results in a class constraint of <Person>. Unifying <Person Name> with <Event> results in <Root Class>.

This strategy has several beneficial results. Features that are merely accidental properties of a particular instance will be dropped quickly. Features that are shared with a similar positive instance are retained. These tend to include essential characteristics of the target concept. The intractable problem of finding an optimal generalization is thus reduced to the simpler problem of finding a similar initial concept definition.

Figure 4.2 shows CRYSTAL's generalization mechanism graphically. The actual instance space has an extremely large number of dimensions, but this two dimensional figure can give some insight into how CRYSTAL operates.

The plus signs are positive instances and the minus signs are negative instances of a target concept in instance space. A concept definition defines a

region in instance space, and is shown as an oval that initially covers only a single seed instance, as shown in Part A of the diagram.

Part B of the diagram shows the first proposed generalization of this initial definition. The current definition covers only the seed instance, shown as a dark boundary. CRYSTAL has relaxed constraints on this current definition just enough to cover the nearest instance. The proposed generalization is shown as a boundary in lighter ink, since CRYSTAL is not yet committed to accepting this definition.

After testing the proposed definition of Part B, CRYSTAL finds that it covers two training instances with no errors. This definition becomes the current definition in Part C of the figure. The next proposed generalization in part C covers four instances with no errors.

In part D of Figure 4.2 CRYSTAL has a current definition that covers four instances with no errors. When this definition is relaxed enough to cover the nearest positive instance will also cover two negative instances.

What CRYSTAL does at this point depends on the error tolerance parameter. The proposed concept definition covers seven training instances with two errors. At error tolerance 0.30, this would be an acceptable definition, and generalization would continue as before. At an error tolerance of 0.20, CRYSTAL would halt generalization. The proposed generalization that exceeds the error tolerance is discarded and CRYSTAL adds the current definition (which covers four instances with no errors) to the rule base.

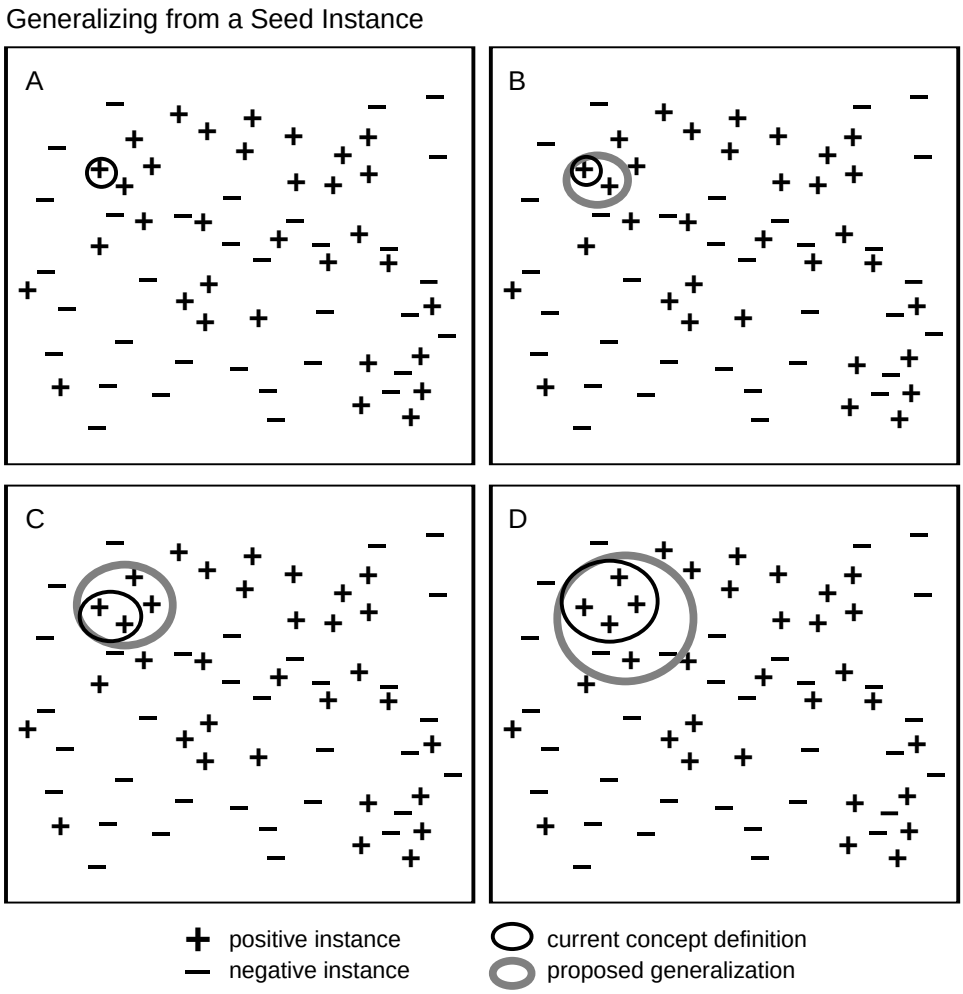

Figure 4.2 Each generalization step relaxes constraints just enough to cover the nearest positive instance.

4.3 Finding Similar Concept Definitions

The similarity metric used by CRYSTAL counts the number of relaxations that would be needed to unify the current concept definition with an initial definition. Dropping one word from a term constraint counts as a single relaxation, as does dropping a mode or root constraint. Moving up one level in the semantic hierarchy counts as one relaxation. Entirely dropping a class constraint is equivalent to the number of relaxations needed to reach the root class.

CRYSTAL applies the similarity metric to all initial concept definitions that extract the same slot values from the same constituents as the current definition. Thus, if CRYSTAL is generalizing a definition that extracts *Person_In* from the subject and *Position* from a prepositional phrase, it will only look for similar definitions that extract these slots from the same constituents.

All constraints on a syntactic constituent are dropped if no corresponding constituent is found in the other definition. Suppose that the current definition has two PP's (prepositional phrases) "as president" and "in 1983" and an initial definition has one PP "as chairman". CRYSTAL matches the most similar PP's, and leaves "in 1983" unmatched[5].

When CRYSTAL looks for the most similar definition, it considers all initial definitions that are not already covered by the current definition. Although initial definitions covered by the existing rule base are not used as seed definitions, they are still available to guide generalization.

[5]When there are more than one of the same-named constituents, such as multiple PP's, CRYSTAL should ideally find a mapping that maximizes the similarity. CRYSTAL approximates this with a greedy assignment based on the similarity between each pair of possibly matching constituents.

CRYSTAL includes a mechanism for biasing the similarity metric to give a greater weight to certain constraints and to certain syntactic constituents than to others. These weights are passed as parameters to CRYSTAL. The settings used in these experiments count relaxation of verb constraints as 1.5 times the weight of other constituents. This weight factor is based on an intuition that verbs tend to be a somewhat more important part of the context than other constituents.

Similarly, the preposition seems to be more important than other words in a prepositional phrase, so it has been given a weight of four times that of other words. No attempt has been made to tune these settings for particular domains, although this could be done. I have experimented very little with the effect of various settings for the distance metric. The impact on system performance seems to be slight. Getting exactly the right similarity metric may not be critical to CRYSTAL's performance, due to the robustness of its search strategy.

4.4 The CRYSTAL Algorithm

Let us now turn to a more formal statement of the CRYSTAL algorithm. CRYSTAL starts with an empty rule base and a set of initial concept definitions, one for each positive training instance. CRYSTAL repeatedly selects as a seed an initial concept definition that is not yet covered by the rule base. A concept definition is generalized from this seed and added to the rule base. This continues until all the initial definitions have either been selected as seeds or are covered by the rule base.

```
The CRYSTAL Algorithm:

Rules = NULL
Derive an initial concept definition from each positive instance
Do for each initial definition D not covered by Rules
    Loop:
        D' = the most similar initial definition to D
        If D' = NULL, exit loop
        U = the unification of D and D'
        Test U on the training set
        If the error rate of U > error tolerance
            Exit loop
        Set D = U
    Add D to the Rules
Return the Rules
```

CRYSTAL belongs to a class of machine learning algorithms known as *covering algorithms*, which find a set of concept descriptions that cover the positive instances and do not cover negative instances. Covering algorithms are discussed more fully in Section 8.3. Where CRYSTAL differs from other covering algorithms such as A^q [Michalski 1983], CN2 [Clark and Niblett 1989], and the candidate elimination algorithm [Mitchell 1978, Mitchell 1982] is in the method used to find generalized concept descriptions.

To generalize a concept definition D, CRYSTAL relaxes the constraints by finding a similar initial definition D' and creating the unification U of D and D'. This process of generalization is repeated until one of two things happens. If no D' can be found, this means that D already covers all possible candidates for similar definition and will not profit from further relaxation.

The more likely situation is that some essential constraint has been relaxed too far in creating the proposed definition U. If the error rate of U exceeds a user-defined error tolerance, then U is discarded. CRYSTAL backs up one

generalization and adds the definition D to the rule base. This is the most general definition found that was within tolerance.

The error rate of a proposed definition is computed by applying the definition to each instance in the training set. These instances have been created from training texts that were hand-annotated to indicate which constituents, if any, contain slot values of the target concept. If all the constraints of a definition are satisfied, but the appropriate slot values are not found, this is counted as an extraction error.

4.5 Robustness of CRYSTAL

Why does CRYSTAL include an error tolerance parameter? This gives CRYSTAL robustness with respect to noisy training data and prevents an otherwise good definition from being blocked by an occasional negative instance. See Sections 8.3.2 and 8.5 for a discussion of the pruning techniques used by the CN2 covering algorithm and by decision tree algorithms to handle noise.

A certain amount of noise tolerance is desirable when processing unrestricted text. One source of noise is the human annotators, who may make errors or be inconsistent in labeling references to the target concepts. In particular, if a phrase is overlooked that should have been marked as a positive instance, this will be considered a negative instance by CRYSTAL. Other sources of noise are parsing mistakes by the sentence analyzer, or semantic tagging that is too coarse to distinguish a negative instance from positive instances.

One unavoidable source of noise comes from the local nature of CRYSTAL instances. Local evidence may not be sufficient to distinguish a relevant from an irrelevant reference. In such cases it may be best to generate rules that sometimes extract negative instances (i.e. irrelevant information). Later processing in an information extraction system can often make use of evidence beyond the local sentence to filter out irrelevant extractions.

For example, a Management Succession text may have a sentence that says "Mr. A will succeed Mr. B" but not mention any company name or corporate post in that sentence. CRYSTAL should create a succession event case frame, even though there is only an 80% probability that this is relevant information for this domain. A previous sentence may indicate that Mr. B was head of a government agency, which makes Mr. A's new job irrelevant to the domain. CRYSTAL is followed by a discourse processing module that can filter CRYSTAL's output based on information outside the local sentence.

CRYSTAL is robust in another way. There is a built-in redundancy in the concept definitions induced by CRYSTAL that tends to overcome sub-optimal choices made during the induction. Once a similar definition has been chosen for unification, CRYSTAL never backtracks to try alternate choices. Unifying with the "most similar" initial definition is not guaranteed to lead to an optimal generalized definition.

It turns out that this is not a serious problem. When CRYSTAL generates a sub-optimal rule, the positive instances that are not covered are available as later seed instances. CRYSTAL continues to add rules until the training instances that were missed earlier have been covered.

Suppose that CRYSTAL is generalizing from a seed instance, and a definition exists that covers 100 positive instances including that seed. What happens if CRYSTAL takes a wrong turn and arrives instead at a definition that covers only 5 of these instances? The other 95 positive instances remain to be chosen as seeds. This gives CRYSTAL 95 more chances to find the high-coverage definition. If it is found, the sub-optimal definition becomes redundant and causes no harm.

Even if CRYSTAL never finds a single definition to cover all 100 instances, CRYSTAL guarantees that the region in instance space containing these instances will still be covered. Several lower-coverage definitions, when taken together, may actually give better performance than a single high-coverage definition. Experiments reported in Section 5.5 show that a compact set of high-coverage rules has a greater tendency to make errors on blind test instances than an overlapping set of lower-coverage rules.

4.6 Time and Space Complexity of CRYSTAL

CRYSTAL has an efficient search control strategy for dealing with instances spaces with extremely large numbers of features. This is important in natural language processing, where the number of word-based features can be extremely large, as discussed in Section 8.2. If a corpus has thousands of distinct words and a feature represents a particular word in one of several syntactic roles, there can be tens of thousands of features.

Let n be the total number of training instances, p be the number of positive instances. Finding a similar definition requires inspecting up to p initial

definitions. The proposed generalization is then tested on n instances. This gives computation time of $O(p+n) = O(n)$ for each generalization step.

Let k be the average number of generalizations per seed, which turns out to be small enough that it can be treated as a constant ($k \approx 3$ for the Hospital Discharge training sets). The number of times CRYSTAL selects a seed to generalize is equal to r, the number of rules generated. This gives time complexity of $O(rn)$.

Ideally r depends only on the underlying concept being learned and is independent of n and p. With noisy data, however, a seed is selected $O(p)$ times. This gives time complexity of $O(pn)$.

The above analysis is for learning a single concept. If a domain has several concepts, the time required for each is $O(pn)$, where n is the same for all concepts and p is the number of positive instances of a given concept.

Note that the size of the feature set does not enter into the time or space requirements of CRYSTAL. The space required is proportionate to the number of instances. Each instance is represented only in terms of the words and semantic classes that actually appear in the instance. There may be thousands of words and hundreds of semantic classes that do *not* appear in the subject, thousands of words that do not appear in the direct object, thousands that do not appear as the object of the preposition "with", and so forth.

It is the bottom up nature of CRYSTAL that allows it to focus on a small number of features at a time. This will also be true of other machine learning algorithms[6] that operate bottom up, such as instance based learning

[6]given an appropriate implementation

[Aha *et al.* 1991, Cost and Salzberg 1993] and some covering algorithms that start from "seed" instances [Michalski 1983].

Algorithms that operate from the top down, on the other hand, typically include a step that considers features exhaustively[7]. This is the case with top down decision tree induction [Quinlan 1993] or decision list induction [Pagallo and Haussler 1990], and some covering algorithms that operate top down [Clark and Niblett 1989]. Extremely large feature sets can have a serious impact on computation time for these algorithms.

Mitchell's candidate elimination algorithm combines aspects of top down and bottom up processing [Mitchell 1978, Mitchell 1982]. The number of features considered in a basic step of candidate elimination is small at first, but may grow quite large during the course of induction.

4.7 A Walk-through Example

This section presents a trace in which CRYSTAL generalizes from a seed instance that has a management succession event with a *Person_In* slot and a *Person_Out* slot. The training set used in this example was from 359 Management Succession texts (60% of the total corpus, selected randomly), which produced 10,570 training instances. Of these 112 were positive instances of a succession event including both a *Person_In* and a *Person_Out*.

Let us take as a seed the initial definition that has already been presented in Figure 4.1. The most similar initial concept definition is shown in Figure

[7] An exception to this is a system by Yali Amit, Donald Geman, and Ken Wilder that uses decision trees for optical character recognition [Amit *et al.* 1995]. It handles an extremely large set of features by considering randomly selected subsets of the features. Multiple trees were built in this way and allowed to vote on the classification.

4.4. For the sake of clarity, I have shown just the motivating sentences for these concept definitions in Figure 4.3.

Seed:
He succeeds Jack Harper, a company founder
who was named chairman.

Most similar:
He succeeds Delbert W. Yocam, a longtime Apple executive
who has held many posts at Apple, including chief operating officer.

Figure 4.3 The input sentences for a seed and the most similar instance.

CRYSTAL found this most similar initial definition by computing the similarity between the seed and each initial definition that extracts *Person_In* from the subject and *Person_Out* from the direct object. The most similar initial definition in Figure 4.4 has much in common with the seed. They have identical subjects and identical verbs. The direct objects of each include a <Person Name> and <Organization>. The direct object has an appositive[8] in both cases. Each sentence ends in a relative clause attached to the direct object that contains the class <Corporate Post> and <Past>.

Unifying the seed with this instance produces the generalized concept definition shown in Figure 4.5. CRYSTAL tests this new definition on the entire training set and finds that it covers two instances with no errors.

This may not seem to be much progress, but it actually represents sixteen individual relaxations on the seed definition: twelve terms, head terms, or modifier terms dropped, and four classes, head classes, or modifier classes have been relaxed one level. <Generic Organization> has been relaxed to

[8] An appositive is two juxtaposed noun phrases, such as "Jack Harper, a company founder".

Concept type: Succession Event
Constraints:
SUBJ::
Terms: HE
Head terms: HE
Mod. terms: <null>
Classes: <Generic Person>
Head classes: <Generic Person>
Mode: affirmative
Extract: Person_In
VERB::
Terms: SUCCEEDS
Head terms: SUCCEEDS
Mod. terms: <null>
Root: SUCCEED
Mode: active, affirmative
OBJ::
Terms: DELBERT_W._YOCAM %COMMA% A LONGTIME APPLE EXECUTIVE
Head terms: DELBERT_W._YOCAM EXECUTIVE
Mod. terms: %COMMA% A LONGTIME APPLE
Classes: <Person Name>, <Generic Post>, <Organization Name>
Head classes: <Person Name>, <Generic Post>
Mod. classes: <Organization Name>
Mode: affirmative
Extract: Person_Out
REL-OBJ::
Terms: WHO HAS HELD MANY POSTS AT APPLE %COMMA% INCLUDING CHIEF_OPERATING_OFFICER %PERIOD%
Head terms: CHIEF_OPERATING_OFFICER
Mod. terms: WHO HAS HELD MANY POSTS AT APPLE %COMMA% INCLUDING %PERIOD%
Classes: <Past>, <Event>, <Corporate Post>, <Generic Post>, <Organization Name>
Head classes: <Corporate Post>
Mod. classes: <Past>, <Event>, <Generic Post>, <Organization Name>
Mode: affirmative

Figure 4.4 Most similar initial concept definition to the seed definition

Concept type: Succession Event
Constraints:

SUBJ::	
Terms include:	HE
Head terms include:	HE
Modifier terms include:	<null>
Classes include:	<Generic Person>
Head classes include:	<Generic Person>
Mode:	affirmative
Extract:	Person_In
VERB::	
Terms include:	SUCCEEDS
Head terms include:	SUCCEEDS
Modifier terms include:	<null>
Root:	SUCCEED
Mode:	active, affirmative
OBJ::	
Terms include:	%COMMA% A
Modifier terms include:	%COMMA% A
Classes include:	<Person Name>, <Organization>
Head classes include:	<Person Name>
Modifier classes include:	<Organization>
Mode:	affirmative
Extract:	Person_Out
REL-OBJ::	
Terms include:	WHO %PERIOD%
Modifier terms include:	WHO %PERIOD%
Classes include:	<Past>, <Corporate Post>
Head classes include:	<Corporate Post>
Modifier classes include:	<Past>
Mode:	affirmative

Covers 2 training instances with 0 errors

Figure 4.5 Unification of "He succeeds Jack Harper, a company founder" and "He succeeds Delbert W. Yocam, a longtime Apple executive who has held many posts at Apple, including chief operating officer"

<Organization>, and <Generic Person> relaxed to <Person>, which is redundant in the presence of a constraint requiring <Person Name>.

If any fewer than these sixteen relaxations had been made, the definition would still cover only the seed instance. A search control mechanism that considered all combinations of single relaxations, then pairs of relaxations, and so forth, would founder before it made enough relaxations to observe any progress. The seed definition has 46 single constraints, $\frac{46 \cdot 45}{2}$ pairs of constraints, and $\frac{46 \cdot 45 \cdot \ldots \cdot 31}{2 \cdot 3 \cdot \ldots \cdot 16}$ combinations of sixteen constraints.

Since the definition in Figure 4.5 has error rate of zero, CRYSTAL continues the generalization. After unifying with "He succeeds William F. Murdoch, 58, who takes the title of vice chairman." CRYSTAL arrives at the definition in Figure 4.6. This has dropped the constraint requiring an <Organization> in the direct object and <Past> in the relative clause. There are no changes to constraints on the subject and verb. This definition covers four training instances with no errors.

Note that this definition still requires the class <Corporate Office> in the relative clause. This class has been engineered to include only job titles that are considered relevant to the Management Succession domain. The next relaxation unifies with "He succeeds Mark Stephens, 48, who resigned in May." This no longer includes the class <Corporate Post>, and lacks any reliable cues that distinguish it from irrelevant instances such as succeeding to a government post or on a board of directors.

After dropping the <Corporate Post> constraint, the definition covers six training instances with one error, giving it an error rate of 0.167 on the training

Concept type: Succession Event
Constraints:

SUBJ::	
Terms include:	HE
Head terms include:	HE
Modifier terms include:	<null>
Classes include:	<Generic Person>
Head classes include:	<Generic Person>
Mode:	affirmative
Extract:	Person_In
VERB::	
Terms include:	SUCCEEDS
Head terms include:	SUCCEEDS
Modifier terms include:	<null>
Root:	SUCCEED
Mode:	active, affirmative
OBJ::	
Terms include:	%COMMA%
Modifier terms include:	%COMMA%
Classes include:	<Person Name>
Head classes include:	<Person Name>
Mode:	affirmative
Extract:	Person_Out
REL-OBJ::	
Terms include:	WHO %PERIOD%
Modifier terms include:	WHO %PERIOD%
Classes include:	<Corporate Post>
Head classes include:	<Corporate Post>
Mode:	affirmative

Covers 4 training instances with 0 errors

Figure 4.6 Class constraints for <Organization> in OBJ and <Past> in REL-OBJ have been dropped after unifying with "He succeeds William F. Murdoch, 58, who takes the title of vice chairman"

data. If the error tolerance has been set at 0.00, CRYSTAL halts and adds the definition in Figure 4.6 with coverage 4 and no errors to the rule base.

At an error tolerance of 0.25, CRYSTAL will continue generalizing and eventually arrive at the definition in Figure 4.7. This covers 72 training instances with 14 errors for an error rate of 0.194. The definition no longer requires a relative clause, has dropped all term constraints, and has even dropped the requirement of a <Person> in the direct object and dropped the requirement that <Person> be the head term of the subject.

```
Concept type: Succession Event
Constraints:
   SUBJ::
      Classes include:        <Person>
      Head classes include:   <Entity>
      Mode:                   affirmative
      Extract:                Person_In
   VERB::
      Root:                   SUCCEED
      Mode:                   active, affirmative
   OBJ::
      Classes include:        <Entity>
      Head classes include:   <Entity>
      Mode:                   affirmative
      Extract:                Person_Out

Covers 72 training instances with 14 errors
```

Figure 4.7 CRYSTAL finally reaches a high coverage definition

If the error tolerance is set at 0.20, CRYSTAL will still find this generalization, but not directly from the first seed. Generalization from this seed halts at a lower-coverage definition with error rate 0.214. One of the positive

instances not covered by this sub-optimal definition is later chosen as a seed and produces the definition in Figure 4.7 after all.

Which error tolerance gives the best performance on unseen texts? Rule sets created at tolerance 0.00, 0.10, 0.20, and 0.25 were tested on a blind set of 240 texts that consisted of 6,106 instances. Even with error tolerance 0.00, CRYSTAL generates rules that are able to identify more than half of the positive test instances of this concept.

As the error tolerance is raised, the number of rules generated decreases, since the average coverage of each rule is greater. At tolerance 0.25, CRYSTAL produces 15 rules that find nearly 80% of the positive test instances, with some decrease in precision.

Performance is measured in terms of *recall* and *precision*, where recall is the percentage of positive instances that were extracted by the rule base. Precision measures the percent correct of instances extracted by the rule base[9].

Tolerance	# Rules	Recall	Precision
0.00	41	53.7	87.8
0.10	34	61.2	87.2
0.20	24	77.6	75.4
0.25	15	79.1	74.6

Table 4.1 Recall and precision for *Person_In,Person_Out* rules at various error tolerance settings

The following chapter presents empirical results for CRYSTAL in both the Management Succession domain and the Hospital Discharge domain.

[9]See Section 5.1 for a discussion of recall and precision.

CHAPTER 5

EMPIRICAL RESULTS IN TWO DOMAINS

5.1 Methodology and Performance Metrics

Experiments in both the Management Succession domain and the Hospital Discharge domain were conducted by partitioning the hand-annotated corpus into a randomly chosen training set and a blind test set. The Management Succession corpus has 599 texts with 16,325 instances. The Hospital Discharge domain has 502 texts with 17,500 instances.

CRYSTAL used the training data to induce a set of concept definitions, which were then applied to the blind test set. The case frames produced by the concept definitions were compared to the hand annotations in the test instances. If every slot contained the desired information, the extraction was counted as correct, otherwise as an error.

The 599 Management Succession texts were randomly partitioned with 119 as training, with 239 as training, and with 479 as training for experiments that show the effect of training size. Other experiments are trained on 239 texts. The 502 Hospital Discharge texts are similarly partitioned with 50, 150, and 351 texts as training for the learning curve experiments and with 251 as training for all other experiments.

Except as noted, all results are averages of ten random partitions of the texts. When statistical significance is mentioned, a two-tailed, paired t-test is

used with $p<0.05$. Experiments that involve hand-engineered semantic tagging or hand-coded rules were done on a single partition of the corpus. The test set was not consulted during the knowledge engineering.

Performance is measured in terms of *recall* and *precision*. Recall is the percentage of positive instances of the target concept that were correctly identified. Precision is the percentage of extractions made that were correct. These metrics are defined more formally in Figure 5.1.

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

	Reported as	Actually
TP (true pos)	pos	pos
FN (false neg)	neg	pos
TN (true neg)	neg	neg
FP (false pos)	pos	neg

Figure 5.1 Definitions of Recall, Precision, and Accuracy

Recall and precision are more useful metrics than *accuracy* when there is an extremely unbalanced distribution of positive and negative instances. With a data set that is 99% negative, it is trivially easy to get accuracy of 99%. Just classify all the instances as negative. Recall and precision focus on how well the system does with the positive instances, which are the ones that matter.

Instances for text extraction typically have only a tiny percent positive. For the Management Succession domain, instances ranged from 99% negative to over 99.9% negative depending on the combination of slots being extracted.[1] Hospital Discharge extraction ranged from 90% negative to over 99% negative

[1]Instances are positive or negative with respect to extraction of particular slots from particular constituents. For example, less than 1% of the Management Succession instances are positive instances of a *Person_In* in the subject and a *Position* in the direct object.

instances. Figure 5.2 compares the accuracy metric with recall and precision for an instance space with 10,000 instances that is 99% negative instances.

	TP	FN	TN	FP	Rec	Pre	Acc
1.	0	100	9,900	0	0	-	99.0
2.	1	99	9,801	99	1	1	98.0
3.	80	20	9,820	80	80	50	99.0
4.	60	40	9,880	20	60	75	99.4

Figure 5.2 Examples of Recall, Precision, and Accuracy

Figure 5.2 shows performance from four hypothetical systems. The first uses the dominant class as a default, and extracts nothing. Recall is zero and precision is undefined (zero divided by zero). Even though this is a totally useless information extraction system it has an impressive accuracy of 99.0%. The second system randomly assigns one percent of the instances as positive. Recall is 1%, precision is 1%, and accuracy is 98.0%.

The next two have more realistic performance. System three correctly identifies 80 out the 100 positive instances. This gives recall of 80%. The system also has 80 false positives, giving it precision of 50%. This would be an extremely useful system for some purposes. A user would need to review only 160 extractions rather than read all the documents corresponding to 10,000 instances. The accuracy for this system, however is identical to that of the system that extracts nothing. System four has recall of 60% (identifies 60 out of 100 positive instances) and has precision of 75% (60 correct out of 80 extractions). Accuracy is 99.4%.

For most of the experiments, the error tolerance parameter is set at 0.20, which tends to give roughly balanced recall and precision on these data sets. Concept definitions that cover only one training instance have been discarded,

since they are too tightly constrained to cover test instances and have a negligible effect on performance.

5.2 Assessing Performance in Information Extraction

When presented with empirical results for an information extraction system, the question arises, "How close to optimal are these results?" It is unrealistic to expect perfect performance from an automated system. Even humans fall considerably short of perfect performance on an information extraction task.

Performance is measured by comparing a system's output with a hand-annotated answer key for the text. Any differences from the answer key are counted as errors. Unfortunately, it is not always clear exactly what the "correct" output should be for a text. Texts often contain ambiguous references that are open to a variety of interpretations.

An experiment [Will 1993] was conducted that measured the amount of inter-coder disagreement between four human analysts. These were professional analysts, each with years of experience at manual information extraction, who annotated the collection of training texts for one of the Fifth Message Understanding Conference domains [MUC-5 1993]. This was a domain of news stories about companies involved in microelectronic chip manufacturing.

The experiment had pairs of analysts read the same texts and independently create output representing the relevant information. One analyst's output was evaluated by treating another analyst's output as the answer key. Different analysts' output for the same text agreed less than 80% on the average. The average recall was 77% at average precision of 79%.

Keep in mind that these are skilled human analysts doing their most careful job. Each of the analysts can be assumed to fully "understand" the text, but is penalized by a limitation built into the scoring process. Real text is open to a variety of reasonable interpretations, which makes formal evaluation by an answer key somewhat arbitrary.

While this is only one rather small experiment in a particular domain, it points to a ceiling on text analysis performance, whether by humans or automated systems. These results are consistent with human performance in previous Message Understanding Conferences. The conference organizer, Beth Sundheim, estimated that the upper limit on human performance for the MUC-4 task was 75% recall and 85% precision [Sundheim 1992].

Not surprisingly, automated systems have somewhat lower performance. The best automated systems in MUC conferences have had recall and precision between 50% and 60% [MUC-3 1991, MUC-4 1992, MUC-5 1993, MUC-6 1995]. Despite different domains and some differences in scoring metrics, there seems to be a ceiling of about 60% recall and precision for current information processing technology.

CRYSTAL is one component of an information extraction system and cannot be directly compared to performance of a full system. In some respects performance of a module that applies text analysis rules should be higher than that of a full system. An extraction by CRYSTAL is counted as correct if it identifies the syntactic constituent (e.g. subject, direct object, or prepositional phrase) containing the relevant information. Such a constituent will typically contain extraneous words that must be discarded for a full system to be judged correct. The full system also faces the difficult task of merging together related

information from separate sentences, and be penalized for any errors made in merging.

On the other hand, formal scoring is *more* strict in some respects for CRYSTAL than for a full system. Multiple references to the same information are often found in a text. CRYSTAL's recall is penalized unless it finds all of the redundant references. A full system merges redundant references into a single output and has full recall even if some redundant references were missed.

A module that applies text analysis rules takes its input from previous automated steps in text analysis must also cope with noisy input. Errors in syntactic analysis are inevitable. If an abbreviation ends with a period that also marks the end of a sentence, an automated sentence analyzer might miss a sentence boundary. If a part-of-speech tagger does not recognize an infrequently occurring verb, it may be taken as a noun and the sentence will be badly mis-analyzed. If a word is not found in a semantic lexicon it will not be given its proper semantic class. An automated semantic tagger will often choose the wrong semantic class for ambiguous words.

Researchers familiar with the current state of information extraction technology would be pleased by recall and precision of 60% for a module such as CRYSTAL that applies text analysis rules, and delighted by recall and precision of 80%. The question of optimal results that was posed at the start of this section might be rephrased as follows. How close does CRYSTAL come to the best possible performance given noisy training data and imperfect evaluation criteria?

This chapter includes two experiments that address this question. Section 5.4 compares CRYSTAL with hand-coded rules that operate on the same input

data and are developed from the same training set as CRYSTAL. Section 5.5 shows results of a beam search version of CRYSTAL that increases the amount of search for optimal generalizations from each seed.

5.3 Empirical Results and Learning Curves

Results will be shown for the Management Succession domain and for two versions of input for the Hospital Discharge domain. The second version is after semantic class assignment of individual words has been customized for the information extraction task.

5.3.1 Management Succession Performance

The *Succession Event* case frame has four slots, *Person_In*, *Person_Out*, *Position*, and *Organization* (abbreviated in performance graphs as *In*, *Out*, *Post*, and *Org*). Since not all instances have all possible slots, CRYSTAL learns each of the fifteen possible combinations of one, two, three, or four slot concepts.

The graphs in Figure 5.3 show recall and precision for each of these concepts. The number of positive training instances is shown for each concept.

CRYSTAL is able to achieve recall and precision in the 60's and 70's for the single-slot concept *Person_In*, for *Person_Out*, for *Position*, and for the two-slot concepts that combine these slots. Performance was a little lower for *Organization* and for multi-slot concepts that include an *Organization*.

Multi-slot concepts have a smaller number of positive training instances than single slot concepts. For example, there are 505 training instances of

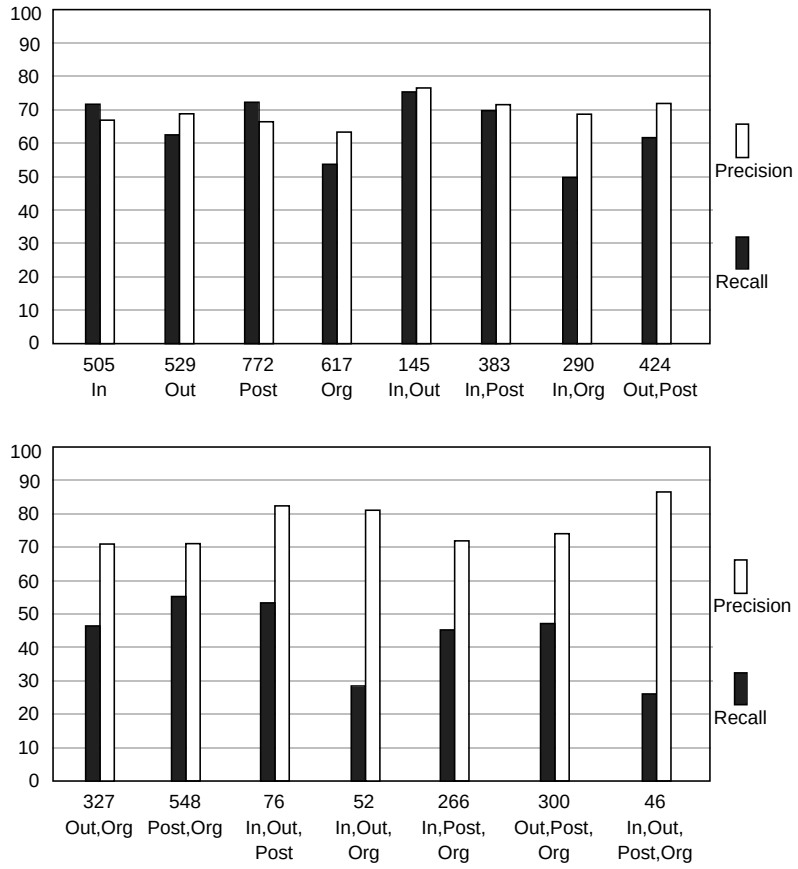

Figure 5.3 Management Succession performance: averages of 10 random partitions into 479 training texts and 120 blind test texts

Person_In, but only 52 of them are part of a *Succession Event* that has *Person_In* together with *Person_Out* and *Organization*.

Was the amount of training adequate for these concepts? Would CRYSTAL's performance increase if more texts were annotated? The learning curves shown in Figure 5.4 give an indication that CRYSTAL had not reached a ceiling in performance at this level of training. Another doubling of the number of training texts should produce another increase in performance. At some point, diminishing returns set in, and further training will have little effect.

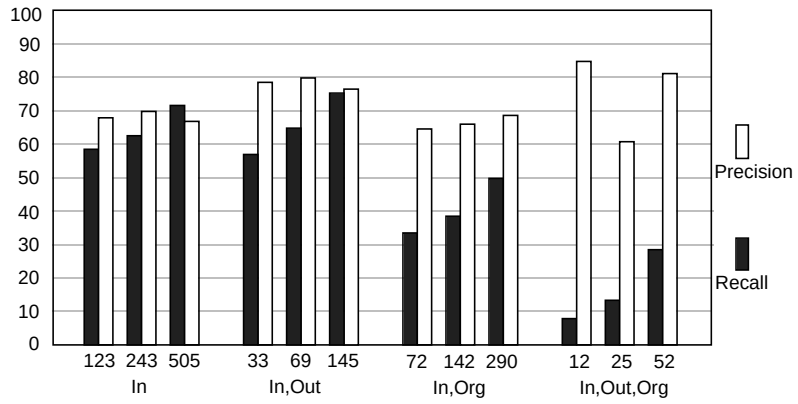

Figure 5.4 Learning curves for the Management Succession domain

Recall and precision are shown for four representative Management Succession concepts[2] as training size is increased from 20% of the texts to 40% of the texts to 80% of the texts. The number of positive training instances is shown below each set of recall and precision.

Recall increases with each doubling of the training size, while there is no significant difference in precision[3]. This has the biggest impact on the multi-slot concepts, which seem to be under-trained. Precision remains fairly level as an artifact of the error tolerance parameter. The error tolerance was kept at 0.20, which would result in precision of about 80 if error rates on the test set exactly mirrored error rates on the training.

Table 5.1 shows the number of concept definitions generated from one of the partitions of 479 training texts. These are broken down by how many training instances they cover. Note that the concepts with high coverage

[2] A table with learning curve results for all fifteen combinations of slots is given in Appendix A.

[3] The difference in recall from 40% training to 80% was statistically significant in every case. The recall had wider variance at lower training levels, which made some of the difference from 20% training to 40% only marginally significant. None of the differences in precision is significant.

Concept	Definitions with coverage:				
	2-4	5-9	10-49	50+	Total
In	62	29	26	8	125
Out	75	37	31	2	145
Post	62	62	43	8	175
Org	99	59	30	0	198
In,Out	6	10	12	1	29
In,Post	49	12	18	4	83
In,Org	52	16	16	0	84
Out,Post	66	30	18	0	114
Out,Org	55	28	12	0	95
Post,Org	66	56	35	0	157
In,Out,Post	7	2	4	0	13
In,Out,Org	9	5	1	0	15
In,Post,Org	49	16	10	0	75
Out,Post,Org	40	22	11	0	73
In,Out,Post,Org	6	3	1	0	10

Table 5.1 Number of concept nodes for Management Succession, broken down by coverage

definitions (covering 50 or more training instances) are those with the highest performance as shown in Figure 5.3.

5.3.2 Hospital Discharge Performance

Experiments were also run on the Hospital Discharge domain, which has four single-slot concepts: *Symptom,Present* and *Symptom,Absent*; *Diagnosis,Confirmed* and *Diagnosis,Ruled_Out*. Figure 5.5 shows results for each of these concepts at three different training sizes[4]. This first set of results uses semantic class assignment of individual words based on a medical thesaurus that was not customized to make distinctions needed for this domain[5]. Results with fine-tuned semantic tagging are presented later in this section.

[4] These results are listed in a table in Appendix B.

[5] The Unified Medical Language System thesaurus of the National Library of Medicine [Lindberg *et al.* 1993].

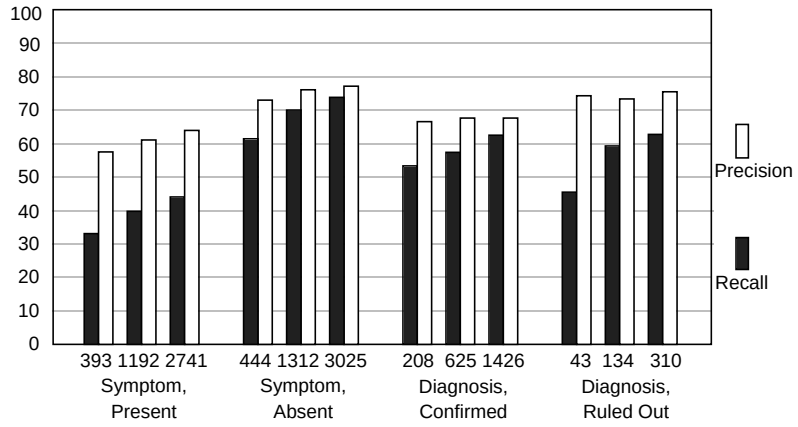

Figure 5.5 Learning curves for the Hospital Discharge domain

As in the Management Succession domain, recall increases with further training, while precision stays fairly flat[6]. It appears that recall for *Symptom,Absent* may be starting to level off at three thousand training instances and *Diagnosis,Ruled_Out* at three hundred.

The number of concept definitions generated from a partition with 351 training texts is shown in Table 5.2. A large proportion of the definitions cover fewer than five training instances, but these low coverage definitions contribute little to performance. Rules for *Symptom,Present* generated from these 351 texts were tested on the remaining 151 texts. When all rules with coverage ≥ 2 are used, recall is 44.1 at precision 65.0. Using only rules with coverage ≥ 5 gives recall 42.2 at precision 69.5. The low coverage definitions contribute little to recall that is not also covered by higher coverage definitions. They also contribute more errors than correct extractions from the test set, thus lowering precision.

[6]All differences in recall are statistically significant. The differences in precision for *Symptom,Present* and *Symptom,Absent* are significant, but not for the other two concepts.

Concept	Definitions with coverage:					
	2-4	5-9	10-49	50-99	100+	Total
Symptom,Present	643	206	83	6	4	942
Symptom,Absent	263	128	133	23	15	562
Diagnosis,Confirmed	254	71	81	14	5	425
Diagnosis,Ruled_Out	26	8	11	6	0	51

Table 5.2 Number of concept nodes for Hospital Discharge, broken down by coverage

As the amount of training increases, a larger proportion of the recall comes from higher coverage definitions. Figure 5.6 shows performance for Hospital discharge rules when rules that cover fewer than ten training instances are discarded.

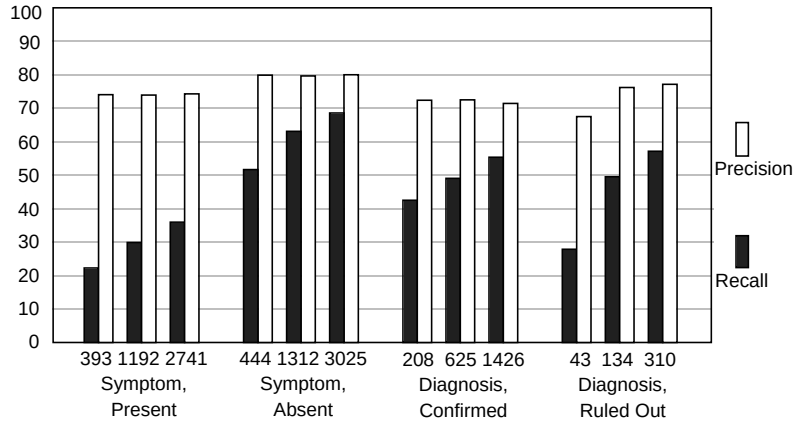

Figure 5.6 Only rules that cover ten or more training instances have been used in this Hospital Discharge learning curve.

With the low coverage definitions removed, precision is closer to the ideal of 80 that would be expected at error tolerance 0.20[7]. A comparison with Figure 5.5 shows that a growing proportion of the recall comes from definitions that cover at least ten training instances. When over 90% of the recall comes from higher coverage definitions, as is the case for *Symptom,Absent* and

[7] All of the differences in recall and none of the differences in precision are statistically significant

Diagnosis,Ruled_Out, this is another indication that the level of training is approaching the point at which more training will increase performance only slightly.

5.3.3 Fine-tuned Semantic Tagging for Hospital Discharge

It should be pointed out that CRYSTAL is faced with considerable noise in the data for this domain. This makes it difficult to find features that characterize the target concepts reliably (hand-coded rules face the same difficultly, as will be discussed in the following section). Noisy data is inevitable when dealing with automatic analysis of unrestricted text. The syntactic analysis and the semantic class assignment will make errors or may be too coarse to make the necessary distinctions. Human annotators also make errors in marking the training texts and produce inconsistent training data.

One source of noise that I was partially able to control was the coarse fit between the semantic tagging of individual words and the information extraction task. A generic medical thesaurus, the Unified Medical Language System (UMLS) [Lindberg *et al.* 1993] had been used with only minor customization.

Unfortunately, class assignment based on UMLS did not correlate closely with annotations of the target concepts. In particular, the class <Sign or Symptom> was a poor predictor of the concept *Symptom,Present*. Only 27% of the phrases annotated as *Symptom,Present* contained a word with class <Sign or Symptom>, and only 71% of the affirmative phrases with that class were annotated as *Symptom,Present*. This is equivalent to recall of 27 and precision of 71.

For the experiment with "fine-tuned" semantic tagging, I modified the semantic hierarchy to make distinctions useful for the Hospital Discharge domain. I set aside half of the corpus as a blind test set and used the remaining 251 texts as a training set. I then tabulated how often each term in the training was associated with annotations for each of the target concepts.

The semantic class assignments were modified according to a term's correlation with *Symptom,Present* or *Symptom,Absent* or with *Diagnosis*. The same modifications were also made on the blind test set.

After fine-tuning the semantic tagging, performance increases as shown in Figure 5.7. Recall and precision are now in the 60's to 80's for all four concepts, with *Symptom,Present* still lagging behind the others.

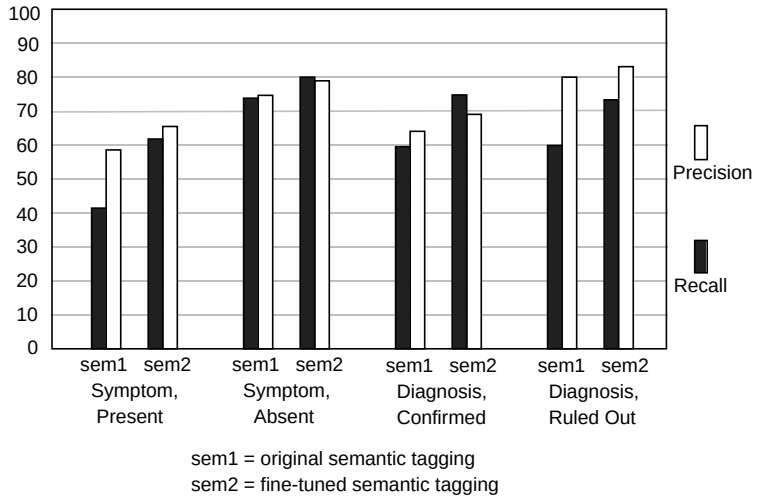

Figure 5.7 Hospital Discharge results with fine-tuned semantic tagging

With less noise in the training instances, CRYSTAL was able to achieve this increase in performance with a more compact set of rules, as shown in Table 5.3. Each concept has less than half as many concept definitions as in

Concept	Definitions with coverage:					
	2-4	5-9	10-49	50-99	100+	Total
Symptom,Present	203	83	79	11	13	389
Symptom,Absent	90	66	67	18	16	257
Diagnosis,Confirmed	65	24	27	6	13	135
Diagnosis,Ruled_Out	9	4	4	6	0	23

Table 5.3 Number of concept nodes for Hospital Discharge after semantic tagging has been fine-tuned

Table 5.2, and the proportion of high-coverage definitions is also greater when noise in the training has been reduced.

5.4 Comparison with Hand-coded Rules

The question still remains of how close to optimal are the rules CRYSTAL learns, given limited and imperfect training data and imperfect evaluation criteria as discussed in Section 5.2. This section presents experiments that compare CRYSTAL with hand-coded rules in an effort to address this question.

I created rules by hand for the Hospital Discharge domain, one set of rules for input with the original semantic tagging and another set of rules based on fine-tuned semantic tagging. Each set of rules took two weeks, and was done after I had been working with CRYSTAL in this domain for one and a half years. I also created rules by hand for four of the Management Succession concepts.

I believe that I was able to put aside any conflict of interest and make the best set of hand-coded rules I could. Anyone else who might have been available to create the rules would have produced an inferior set of rules. Manual engineering of text analysis rules requires someone who is knowledgeable about the domain and intimately familiar with the information extraction system.

5.4.1 Methodology

Fifty percent of the Hospital Discharge texts were set aside as a blind test set and rules were developed without consulting these texts. For the Management Succession domain, forty percent of the texts were used as a blind test set, and rules were developed from the remaining sixty percent. These test sets were not perfectly blind, since I had been seen all of them during previous experiments with CRYSTAL. Creating reliable text analysis rules is delicate enough that any vague impressions I had from previously seeing the eight or ten thousand test instances was negligible.

The representation for the hand-crafted rules was the same as that of CRYSTAL's rules. The concept definitions for the Hospital Discharge domain were augmented to include exceptions, which explicitly list classes and terms to be excluded. This is described more fully in Section 6.1. Rules for Management Succession were created without exceptions, both for CRYSTAL and hand-coded rules.

The methodology for creating rules by hand was similar to CRYSTAL's. I worked through the training texts, a portion at a time, to find positive instances not covered by existing rules. New rules were created to cover these instances and each new rule was tested on the entire training set. Constraints were relaxed as far as possible while keeping the error rate low.

Functions from CRYSTAL assisted by creating an initial concept definition for each positive instance missed by the rules. I edited these initial definitions to remove constraints that did not seem essential to the target concept. Functions from CRYSTAL then tested these hand-crafted rules against the entire training set, listing the correct extractions and extraction errors for each rule.

After several rounds of refinement, the new rules were added to the rule base and this process was repeated for another portion of the training texts. High-coverage concept definitions were discovered early on. As development continued, the additional definitions tended to have either low coverage or high error rate, until no further useful rules could be found.

5.4.2 The Nature of the Instance Space

This diminishing rate of learning useful rules comes from an inherent characteristic of the training data. Only a portion of the positive instances could be identified by high coverage rules. Before semantic fine-tuning, rules that cover 20 or more training instances accounted for only 25% of the *Symptom,Present* instances, 51% of the *Symptom,Absent*, 61% of the *Diagnosis,Confirmed*, and 42% of the *Diagnosis,Ruled_Out*. After fine-tuning, these percentages are raised to 59%, 71%, 72%, and 60%, respectively.

The remaining positive instances are what I will call the "hard" instances. Many of these have semantic classes that are usually not associated with the target concept and must be identified by more restrictive constraints. The evidence that distinguished these instances as positive was often the occurrence of a low-frequency term or combination of terms.

An example of a hard instance is one in which the phrase "under significant family stress" was marked as a *Symptom,Present*. The semantic class assigned to "stress" is <Pathologic Function>, which was associated with diagnoses more often than with symptoms. This was the only occurrence of "under stress" in the training data, and it was not found at all in the test set.

A example of a hard instance from Management Succession is "... at Riggs National Bank, which brought in Paul Homan as its president and CEO in June." This was the only instance in the training set with *Organization* in a PP (prepositional phrase) and *Person_In* in a REL-PP (relative clause attached to a PP). Many of the hard instances for Management Succession were due to syntactic complexity such as extraction from relative clauses.

When such idiosyncratic positive instances occur only in the blind test set, they will be missed by the rules, whether hand-coded or generated by CRYSTAL. This places a ceiling on the performance possible, given a limited training set. As training size increases, the problem due to low-frequency terms is somewhat abated. A region with only five positive instances may have ten when training is doubled, and a region with only two may have four.

Keep in mind that CRYSTAL can never expect clean training data, since it receives its input from other components of an information extraction system. The components that perform the syntactic bracketing and semantic tagging of unrestricted text are liable to make errors. The texts themselves and the hand-annotations used in training may also have errors. Any of these sources of error can account for some of the instances that are beyond the reach of text analysis rules, whether hand-coded or learned by CRYSTAL.

5.4.3 Results

Table 5.4 compares the performance of CRYSTAL with hand-coded rules both before and after semantic fine-tuning. CRYSTAL used an error tolerance of 0.20 for these inductions. The last column in the table uses the average of

Before semantic fine-tuning:

Concept	CRYSTAL			Hand-coded			Ratio of Avg. R,P
	R	P	Avg	R	P	Avg	
Symptom,Present	41.2	58.5	49.8	51.4	81.6	66.5	75.0
Symptom,Absent	73.8	74.7	74.2	74.7	93.3	84.0	88.4
Diagnosis,Confirmed	59.4	64.0	61.7	69.5	78.5	74.0	83.4
Diagnosis,Ruled_Out	59.8	80.0	69.9	71.8	86.6	79.2	88.3

After semantic fine-tuning:

Concept	CRYSTAL			Hand-coded			Ratio of Avg. R,P
	R	P	Avg	R	P	Avg	
Symptom,Present	61.9	65.6	63.8	64.9	79.3	72.1	88.4
Symptom,Absent	80.0	78.9	79.4	79.6	91.9	85.8	92.6
Diagnosis,Confirmed	74.8	69.1	72.0	76.8	77.5	77.2	93.3
Diagnosis,Ruled_Out	73.5	83.1	78.3	81.2	87.2	84.2	93.0

Table 5.4 A comparison of CRYSTAL to hand-coded rules for Hospital Discharge

recall and precision to compute the ratio of CRYSTAL's performance to that of the hand-coded rules.

CRYSTAL achieves 93% the performance of hand-coded rules for three of the concepts and 88% for the fourth after semantic fine-tuning. Performance is lower for both CRYSTAL and for hand-coded rules in the noisier situation in which semantic tags have not been customized for the information extraction task. CRYSTAL's performance ranges from 75% to 88% that of hand-coded rules. CRYSTAL is able to compensate for noisy training to a certain degree, but is more likely than a human developer to be misled by spurious regularities.

Table 5.5 compares CRYSTAL to hand-coded rules in the Management Succession domain for four concepts that I chose arbitrarily. CRYSTAL achieves over 90% of the performance of hand-coded rules for these four concepts, with performance equal to hand-coded for one of them.

One of the primary advantages a human coder has over CRYSTAL is bringing in outside knowledge to find rules for the hard instances. When the feature

Concept	CRYSTAL			Hand-coded			Ratio of Avg. R,P
	R	P	Avg	R	P	Avg	
Person_In	67.2	66.9	67.0	70.6	75.0	72.8	92.0
Person_In,Person_Out	77.6	75.4	76.5	79.1	80.3	79.7	96.0
Person_In,Position	60.0	71.5	65.8	61.7	86.0	71.8	91.6
Person_In,Organization	52.1	69.5	60.8	47.9	72.0	60.0	101.3

Table 5.5 A comparison of CRYSTAL to hand-coded rules for Management Succession

that makes an instance positive is a feature shared by only a few other training instances, CRYSTAL may find instead an irrelevant feature also shared by positive instances. With sparse training data, there is no margin of error if CRYSTAL relaxes the wrong constraint.

In Table 5.6, rules based on fewer than twenty training instances have been eliminated. CRYSTAL comes closer to human performance in the regions of instance space containing at least twenty positive training instances than it does for the more sparsely represented regions.

Before semantic fine-tuning:

Concept	CRYSTAL			Hand-coded			Ratio of Avg. R,P
	R	P	Avg	R	P	Avg	
Symptom,Present	23.9	78.9	51.4	25.1	92.5	58.8	87.4
Symptom,Absent	64.9	80.4	72.7	51.2	96.1	73.7	98.6
Diagnosis,Confirmed	47.6	70.4	59.0	61.4	81.7	71.5	82.5
Diagnosis,Ruled_Out	44.9	87.5	66.2	41.9	85.2	63.5	104.3

After semantic fine-tuning:

Concept	CRYSTAL			Hand-coded			Ratio of Avg. R,P
	R	P	Avg	R	P	Avg	
Symptom,Present	57.3	74.2	65.8	58.8	83.3	71.0	92.7
Symptom,Absent	75.7	83.0	79.3	71.0	93.9	82.5	96.1
Diagnosis,Confirmed	72.2	73.9	73.1	72.2	77.9	75.1	97.3
Diagnosis,Ruled_Out	67.5	86.3	76.9	59.8	89.7	74.8	102.8

Table 5.6 Using only rules that cover twenty or more training instances for CRYSTAL-generated and hand-coded rules

Another way to assess how well CRYSTAL performs is considered in the next section. If CRYSTAL expends more effort searching for an optimal gen-

eralization from each seed, will this raise the over-all performance of the rule base?

5.5 Beam Search

CRYSTAL is able to navigate so efficiently through a large space of possible concept definitions because of the "greedy" nature of the algorithm. At every step in generalizing a concept definition, CRYSTAL is faced with several choices of constraints to relax. CRYSTAL makes the choice that seem to be the best at the time, even though a different choice may turn out later to have been better.

I tried an alternate approach that is more computationally expensive, but has a greater chance of making optimal choices. A *beam search* tries several paths in a search space in parallel. The amount of search effort is controlled by two parameters, the beam width w and branching factor b. When CRYSTAL generalizes from a seed instance, a *beam set* of size w is maintained. These are the best w generalized concept definitions found so far.

For each definition in the beam set, CRYSTAL finds b distinct relaxations by unifying with the most similar initial definition, the next most similar, and so forth. This produces a list of wb generalized definitions, which is sorted to keep the best w distinct definitions. The metric I used to choose the best definitions is to count the number of positive training instances covered. If two definitions cover the same number of positive instances, the definition that covers fewer negative is considered better.

The CRYSTAL algorithm as described in Section 4.4 is equivalent to a beam search with $b = 1$ and $w = 1$. I ran experiments for the Management

Succession domain and the Hospital Discharge domain at a range of beam sizes. Beam width was set to 1, 2, 5, and 10, with branching factor equal to the beam width.

Figure 5.8 shows results at beam size 1, 2, 5, and 10 for four representative Management Succession concepts. Figure 5.9 shows results of a similar beam search experiment for the Hospital Discharge domain with no semantic fine-tuning. The shaded dot indicates the average of recall and precision.

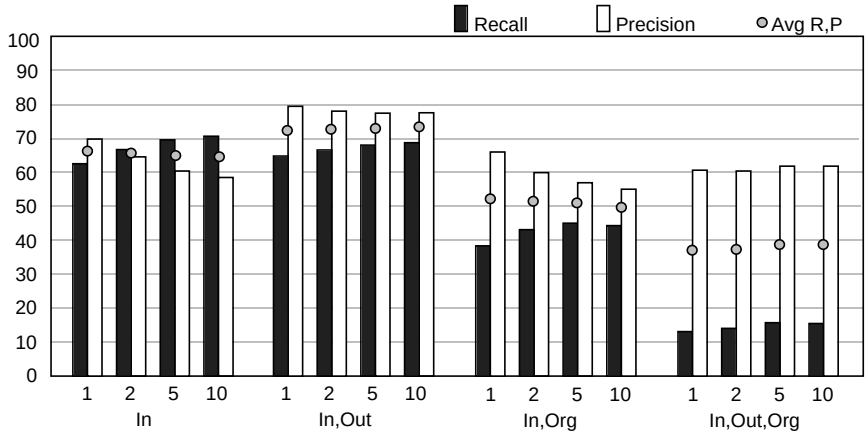

Figure 5.8 Management Succession results at beam width 1, 2, 5, and 10

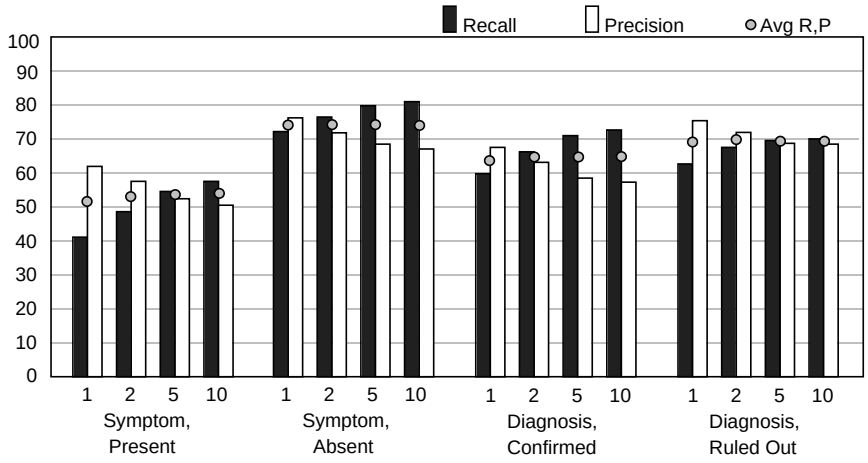

Figure 5.9 Hospital Discharge results at beam width 1, 2, 5, and 10

The total computation time[8] for all four Management Succession concepts averaged 14 minutes at $b = w = 1$, 18 minutes at $b = w = 2$, 36 minutes at $b = w = 5$, and 82 minutes at $b = w = 10$. Computation for the Hospital Discharge concepts took longer, since there were ten times as many positive instances, but a similar ratio held: 1.7 hours at $b = w = 1$, 2.8 hours at $b = w = 2$, 7.1 hours at $b = w = 5$, and 17.3 hours at $b = w = 10$.

Increasing the beam width results in a gain in recall that is almost exactly offset by a drop in precision. The greatest change in recall and precision comes in moving from beam width 1 to beam width 2. There is little effect from moving from beam width 5 to 10. This holds for nearly all concepts in both domains. Most of the changes in recall and precision are statistically significant, but few of the changes in average recall and precision are significant, as shown in Figure 5.10.

	1 - 2			2 - 5			5 - 10		
	R	P	Avg	R	P	Avg	R	P	Avg
In	x	x		x	x		x	x	
In,Out		x							
In,Org	x	x			x				
In,Out,Org									

	1 - 2			2 - 5			5 - 10		
	R	P	Avg	R	P	Avg	R	P	Avg
Symptom,Present	x	x	x	x	x		x	x	x
Symptom,Absent	x	x		x	x		x	x	
Diag,Confirmed	x	x	x	x	x		x	x	
Diag,RuledOut	x	x		x	x				

x = significant at p < 0.05

Figure 5.10 Statistical significance of changes in beam width

The increase in recall is easy to understand. CRYSTAL was searching for reliable generalizations that had the largest coverage possible. More search effort results in rules of greater coverage on the training that are likely to cover more test instances as well.

Why should precision go down? This is a case of a well known phenomenon in machine learning called *overfitting*. A machine learning algorithm may

[8] on a DEC ALPHA AXP 3000

create a concept description that fits accidental characteristics of the training as well as finding features that truly represent the target concept. Overfitting will appear to increase accuracy when measured on the training data, but will actually reduce accuracy on the test set.

Quinlan and Cameron-Jones [95] point out a type of overfitting that results from expending too much effort searching for optimal rules. Among a very large set of possible concept descriptions will be a few "flukes" with high coverage that seem to be highly reliable on the training data, but perform poorly on the test set. A modest amount of search will capture the most salient regularities in the data, but more extensive search is likely to discover a fluke concept description.

Another reason that an increase in recall is offset by a decrease in precision comes from an inherent trade-off between recall and precision. CRYSTAL without beam search is able to find all the positive instances that can be identified reliably. The remaining positive instances are in contexts that are difficult to distinguish from negative instances. Rules that are generalized enough to cover some of these positive instances will also erroneously cover some negative instances. Thus recall is raised at the expense of precision.

5.6 Manipulating a Recall-Precision Trade-off

A graphic illustration of the trade-off between recall and precision is shown in Figure 5.11. In this instance space many of the positive instances (+) are surrounded by negative instances (-). A set of concept definitions can avoid these regions of instance space and maintain high precision at the expense of

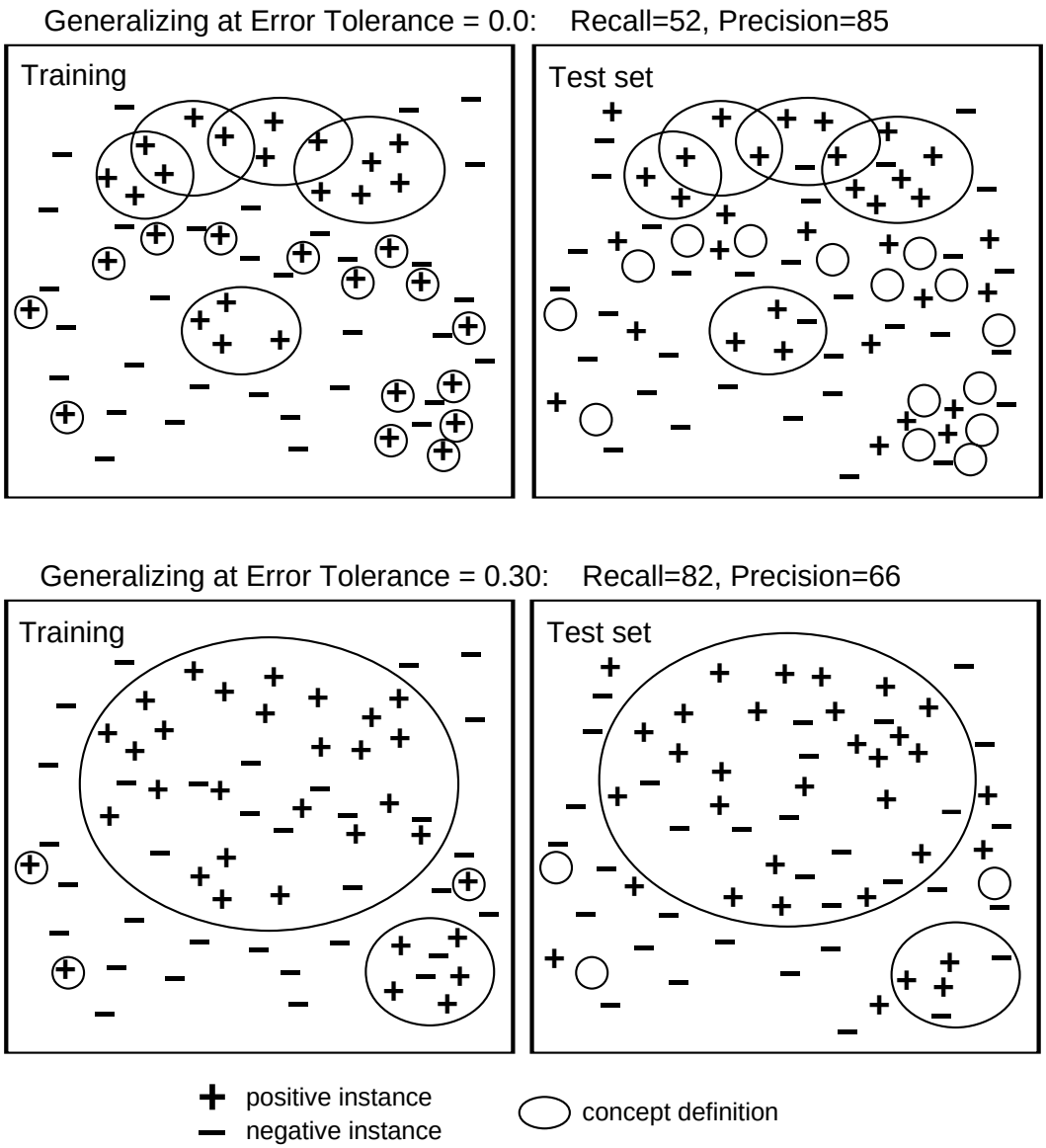

Figure 5.11 An instance space in which high recall or high precision is possible, but not both

recall. An alternate set of concept definitions could gain high recall at the expense of precision.

If error tolerance is set at 0.0, CRYSTAL will find concept definitions that avoid covering any negative training instances, as shown in the upper part of the diagram. This will tend to produce an overlapping set of low-coverage definitions.

With a higher error tolerance of 0.30, CRYSTAL will tend to find concept definitions with high coverage, as shown in the lower half of the diagram. There are still a few isolated positive instances that cannot be generalized without covering the adjacent negative instances.

The right side of Figure 5.11 shows the blind test instances associated with the training instances on the left. The test instances are similar, but not identical to the training set.

The definitions generated at error tolerance 0.0 cover 52% of the positive test instances with precision of 85%. At error tolerance 0.30, the definitions cover 82% of the positive test instances with precision of 66%.

The behavior on test instances illustrated here is typical of a real instance space. Concept definitions that cover a single training instance are so tightly constrained that they are useless on the test set. Low coverage definitions are poor at predicting behavior on blind test instances. A definition that covers four training instances with no errors will frequently cover fewer test instances and will often cover negative as well as positive instances.

In an instance space such as the one in Figure 5.11, the trade-off between recall and precision is inherent to the geometry of the instance space. Many

of the positive instances can only be covered at the expense of covering neighboring negative instances.

In a real instance space, these positive instances that are surrounded by negative instances may be an artifact of the features used to represent instances. Suppose that the semantic class assignment for Management Succession makes an error and fails to recognize a company name and gives it the class <Person Name>. A positive instance of *Organization* with that semantic tag will be indistinguishable from negative instances.

The inability to distinguish positive from negative instances may also be due to limitations in syntactic analysis or in the expressive power of CRYSTAL's concept definitions. A concept definition in the Hospital Discharge domain that correctly identifies "history of cancer" as a *Diagnosis,Confirmed* will also cover "history of cancer in father", which is a negative instance.

CRYSTAL includes two parameters than can be used as knobs to manipulate the trade-off between recall and precision deliberately. One of these is the error tolerance parameter. Increasing the error tolerance allows concept definitions of greater generality, even if they have a greater tendency to cover some negative instances as well as positive. This has the effect of increasing recall at the expense of precision.

Figures 5.12 and 5.13 show the effects of varying the error tolerance in the Management Succession and Hospital Discharge domains. Recall increases and precision decreases for every concept in these two domains as error tolerance goes from 0.0 to 0.40. The gain in one metric is almost exactly compensated by a loss in the other, leaving the average of recall and precision fairly flat. Each

of the differences in recall and in precision shown in both figures is statistically significant[9].

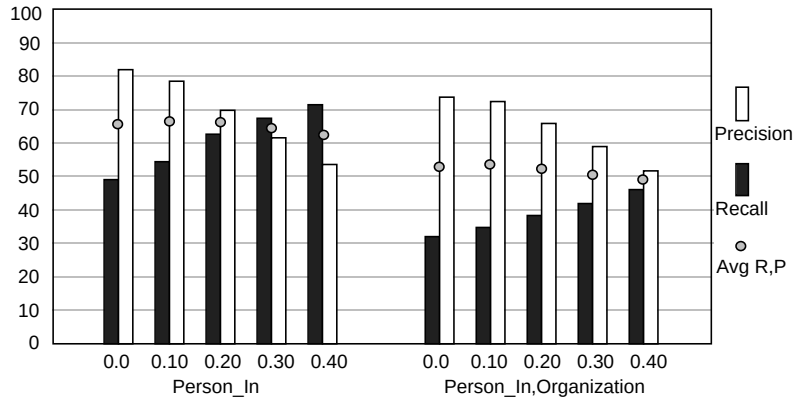

Figure 5.12 Management Succession results at error tolerance 0.0 to 0.40

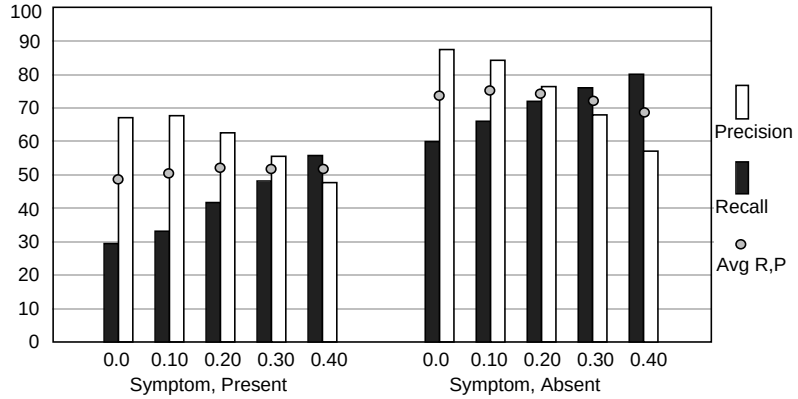

Figure 5.13 Hospital Discharge results at varying error tolerance

Another parameter operates on the completed rule base after induction. The *min-coverage* parameter allows rules to be discarded that do not cover at least a minimum number of training instances. Low coverage definitions tend to be unreliable predictors of performance on the test set. Raising the

[9]The one exception to this is the change in precision from tolerance 0.0 to 0.1 in *Person_In, Organization*. The differences in average recall and precision are small and only a few are significant.

min-coverage has the effect of increasing precision at the expense of lowering recall. This may produce a small increase or small decrease in the average of recall and precision, depending on the concept.

Figure 5.14 shows the interaction of error tolerance and min-coverage[10]. At error tolerance 0.0, precision approaches 100 as min-cover is raised. At error tolerance 0.20, precision approaches 80 or a little above as min-cover is raised. This behavior is true for all concepts in both domains.

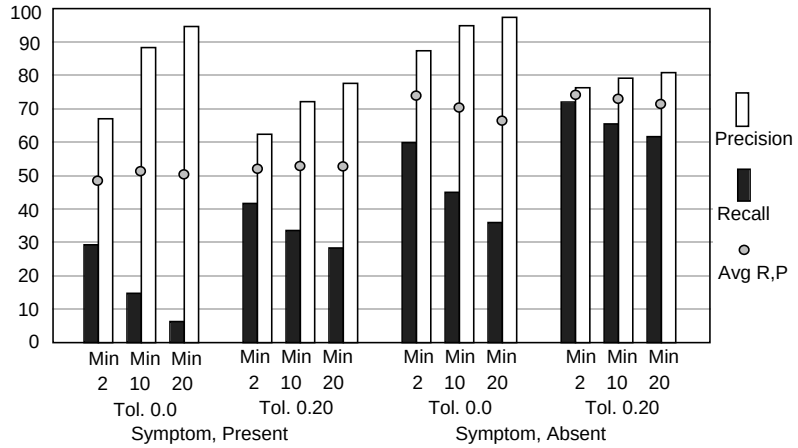

Figure 5.14 Effect of minimum coverage parameter at error tolerance 0.0 and 0.20

The combination of error tolerance and min-cover allow a user to control for the relative importance of recall and precision. For some applications, precision is critical and recall of 36 at precision 97 is better than recall of 72 at precision 76. For other applications the reverse may be true.

[10] All of the changes in recall and in precision are statistically significant. All changes in average recall and precision are significant except for *Symptom,Present* from min-coverage 10 to 20 at tolerance 0.20.

5.7 Run-time Efficiency

The time complexity of CRYSTAL is given in Section 4.6 as $O(pn)$, where p is the number of positive training instances and n is the total number of training instances. Actual computation time is consistent with this analysis.

Table 5.7 shows the CPU time required to generate rules for two representative concepts at different levels of training. This was run on a DEC ALPHA AXP 3000 with 64 Mb of memory. The last column shows the rate of growth of the CPU time[11].

Concept	Positive Insts.	Total Insts.	CPU Seconds	Ratio CPU sec
In	123	3,525	30.1	1.00
	243	6,910	116.2	3.86
	376	10,570	250.4	8.32
	505	13,548	435.7	14.51
In,Out,Org	12	3,525	3.7	1.00
	25	6,910	11.9	3.25
	40	10,570	25.2	6.87
	52	13,548	43.3	11.80

Table 5.7 Growth of computation time as training size increases

The training size in this chart is growing at roughly a ratio of 1:2:3:4, which gives a predicted ratio of 1:4:9:16 for computation time. The actual CPU times for training sizes used in these experiments show a growth rate a bit slower than pn. The concept *In* has about ten times as many positive instances as *In, Out, Org*, and has the predicted ten-fold increase in computation time.

As long as the entire set of training instances fits into memory (as is the case for the training sizes reported in this chapter), clock time closely matches

[11]CPU time was measured with the UNIX command, "time". This is slightly contaminated by CPU cycles used in reading or writing to disk. This is small compared to computation time to induce the concept definitions.

CPU time. If resident memory is exceeded, frequent memory swapping can greatly reduce CPU efficiency unless care is taken with memory allocation.

5.8 Discussion of Results

What conclusions about CRYSTAL's performance can be drawn from the empirical results presented in this chapter? In particular, how close does CRYSTAL come to generating rules with optimal performance, given limited training data and imperfect input, and evaluated against somewhat arbitrary annotations in a test set?

CRYSTAL is able to learn rules from noisy training data that approach the performance of hand-coded rules. Noise can come from several sources when dealing with unrestricted text: inadequate syntactic analysis, inadequate semantic tagging, and inconsistent hand-annotation of the training texts. These sources of noise were reduced, but not eliminated, in the Management Succession data and the fine-tuned Hospital Discharge data.

CRYSTAL achieved an average of recall and precision that was over 90% as high as hand-coded rules for Management Succession, equaling that of the hand-coded rules for one concept. For the fine-tuned Hospital Discharge data, CRYSTAL did 93% as well as hand-coded rules for three concepts and 88% as high for a fourth concept. In a noisier version of the Hospital Discharge data, performance was lower both for CRYSTAL and for hand-coded rules, with CRYSTAL's performance from 75% to 88% that of hand-coded rules.

CRYSTAL's covering algorithm approach gives it robustness in the face of noisy data and hard-to-classify instances. Some regions of instance space may contain such a mix of positive and negative instances, that no single rule will

cover only the positive. CRYSTAL will find a set of tightly constrained rules, each rule covering a portion of the positive instances. Taken together, this set of rules covers all but the most difficult to distinguish positive instances.

CRYSTAL has two parameters that give a user control over how CRYSTAL compensates for noise and for sparse data. Raising the error tolerance parameter increases recall at the expense of precision, while raising the min-coverage parameter has the opposite effect.

Another way to assess CRYSTAL's optimality is to increase the amount of search for an optimal generalization from each seed instance. This turns out merely to raise recall at the expense of precision. Even though CRYSTAL makes sub-optimal choices in generalizing from a particular seed, the positive instances it thus misses become seeds for later generalizations. The aggregate set of rules cannot be improved by more extensive beam search.

The limit to CRYSTAL's performance comes from noise in the data, inadequate features to represent the instances, or insufficient training data. How much training is enough? Performance increases monotonically with more training for the range of training sizes used in these experiments. Diminishing returns eventually set in. The amount of training required depends on the difficulty of the concept being learned.

Diagnosis,Ruled_Out reached recall 60 at precision 73 from only 134 positive training instances. *Symptom,Present* needed 2,741 positive training instances to reach recall 44 at precision 64. A similar disparity held between Management Succession concepts. *Person_In,Person_Out* had recall 65 at precision 80 from only 69 positive instances, while 617 positive instances of *Organization* gave recall of 54 at precision 63.

Creating an annotated training corpus is moderately labor intensive. The Hospital Discharge texts were annotated at a rate of about five per hour, and the Management Succession texts at about twenty per hour. This can be done by an end user with no background in linguistics or computer science. The annotated corpus is not an additional cost over a hand-coded approach. For all but the simplest concepts, an annotated training set is also needed to guide hand-coded rule development.

A system developer can make a greater impact on performance by refining the components that produce CRYSTAL's input than by increasing the amount of training data. A boost in performance results from customizing the semantic class assignment of individual words to the information extraction task. The modest training sizes required by CRYSTAL is due to the power of semantic class representation of the input instances. This allows single rules to cover a large number of instances, and to generalize to words not found in the training set.

CRYSTAL's robustness comes from its rich representation that combines both semantic classes and lexical terms. The following chapter explores the impact of CRYSTAL's rule representation.

CHAPTER 6

IMPACT OF RULE REPRESENTATION

How much of CRYSTAL's performance and robustness in the face of noisy data comes from the flexibility and expressiveness of its rules? This chapter first explores the impact of an enhancement to CRYSTAL's rule representation, then the impact of restricting CRYSTAL's representation in various ways.

6.1 Learning Exceptions to Rules

CRYSTAL's concept definitions are expressed solely in terms of positive constraints: specifying semantic classes or terms that must be found in the instance. There is no provision for a concept definition to specify a word or class that must *not* occur. This can be remedied by adding explicit exceptions to CRYSTAL's representation.

The lack of negative constraints appeared to be a serious limitation, particularly in an earlier version of Hospital Discharge that did not label phrases as affirmative or negative. CRYSTAL had no mechanism to express a concept definition for *Symptom,Present* that excluded instances containing words such as "no" or "not".

The concept definition shown in Figure 6.1 looks for the pattern "revealed <Sign or Symptom>" to identify *Symptom,Present*. It operates correctly on

```
Concept type: Symptom,Present
Constraints:
   VERB::
      Terms include:     REVEALED
      Mode:              active
   OBJ::
      Classes include:   <Sign or Symptom>
      Extract:           Symptom,Present
```

Figure 6.1 An under-constrained definition for *Symptom,Present*

instances such as "Her lungs revealed bibasilar rales" or "A CT revealed a large intra-abdominal mass".

Unfortunately, this concept definition does not exclude instances in which the <Sign or Symptom> is negated. About half the time the definition erroneously applies to negative instances such as "Examination revealed no masses and no axillary lymphadenopathy".

There is less need for exceptions when each phrase in the input has been labeled as affirmative or negative. This allows a concept node to exclude negative phrases by adding a constraint that requires the affirmative mode. Even then, cases still remain in which exceptions to a rule could lower the error rate of a concept definition.

In the Hospital Discharge domain, a definition that looks for "history of <Disease or Syndrome>" is fairly reliable at identifying instances of *Diagnosis,Confirmed*. This definition makes extraction errors, however, when it is a history of disease in a family member rather than in the patient.

Figure 6.2 shows how exceptions can be added to a concept definition to exclude these errors. This definition covers "history of coronary artery

```
Concept type: Diagnosis,Confirmed
Constraints:
   SUBJ::
      Terms include:     HISTORY
      Classes include:   <Disease or Syndrome>
      Mode:              affirmative
      Extract:           Diagnosis,Confirmed
Exceptions:
   Exception:
      SUBJ::
         Terms include:     FAMILY
         Extract:           Diagnosis,Confirmed
   Exception:
      PP::
         Classes include:   <Family Group>
```

Figure 6.2 A concept definition with two exceptions

disease", but does not cover "family history of coronary artery disease" or "history of coronary artery disease in father".

An exception can use any of the constraints used in a concept definition: terms, head terms, modifier terms, root, preposition, classes, head classes, modifier classes, mode. A definition with exceptions applies to an instance if all of the positive constraints and none of the exceptions are satisfied.

Examples in which exceptions are useful can be found in any domain. Section 4.7 presented a definition for *Person_In,Person_Out* in the Management Succession domain that looks for the pattern "<Person> succeeds <Person>". This definition has an error rate of nearly 20%, which could be reduced by exceptions that exclude "succeeds to the board", "succeeds as <Government Position>" and so forth.

In many regions of instance space there will be a predominance of positive instances, but pockets of negative instances. Two approaches are possible to identify the positive instances. CRYSTAL can create several concept definitions without exceptions, each of which covers a relatively small region that avoids the pockets of negative instances. This approach is likely to miss some of the positive instances that are hard to separate from the negative ones.

The alternative is for CRYSTAL to create a single over-generalized definition that covers the entire region, including the negative sub-regions. Exceptions are then added to exclude as many of the negative instances as possible. This approach will tend to have higher recall than the first, but may have lower precision if exceptions fail to exclude all of the negative instances.

6.1.1 An Algorithm for Learning Exceptions

The usefulness of adding exceptions to concept definitions depends on an algorithm for automatically learning exceptions. CRYSTAL with exception learning follows the basic algorithm up to the point at which a proposed generalization is found to exceed error tolerance. At that point, the basic CRYSTAL algorithm would halt generalization and discard the over-generalized definition.

CRYSTAL with exception learning does not halt when it reaches a definition with excessive errors. Instead it identifies features that characterize the negative instances but not the positive instances covered by the definition. If the instances covered by the definition include sub-regions in which negative instances are clustered together, an exception can be added to exclude each of these negative sub-regions. After adding exceptions, CRYSTAL tests the

error rate again and continues generalizing if the definition is now below error tolerance.

Adding exceptions will not always reduce the error rate of the definition sufficiently. The proposed definition before exceptions are added covers a region of instance space with an excessive proportion of negative instances. If these negative instances are randomly scattered throughout the region covered, CRYSTAL will be unable to find exceptions that exclude the negative instances without also excluding the positive. In such a case, CRYSTAL will halt and discard the over-generalized definition.

There is a more subtle situation in which CRYSTAL should abandon a definition rather than add exceptions. It might be possible to find exceptions that are so specific to the training examples that they fail to exclude any negative instances in the test set. In this case the error rate will appear to have been reduced, but will still exceed the error tolerance on previously unseen test instances, which is what really matters.

The region of instances covered by a definition will generally contain a much smaller number of training instances than the entire training set. In this small sample of instances, nearly every negative instance will include some words or even semantic classes that are unique to that instance.

It may be the only instance with the prepositional phrase "since last April". Even though this phrase has nothing to do with making this a negative instance, an exception that excludes instances with "since last April" will reduce the error rate on the training. Such an exception will probably not apply to any test instances covered by the definition and may exclude a positive instance if it does.

To avoid such spurious exceptions, CRYSTAL only considers features found in at least two extraction errors as a candidate for an exception. The feature is tested on all instances covered by the definition. If it is found primarily in negative instances and not positive, it is added as an exception to the definition. A new parameter, the exception tolerance (XTol), may be set greater than 0.00 to allow an exception to exclude a percentage of positive instances as well as negative.

Exclusion of negative instances may have brought the error rate within error tolerance. If so, generalization continues. A second, possibly more restrictive, error tolerance (Tol2) is used for a rule with exceptions. If the exceptions do not reduce errors sufficiently, CRYSTAL halts, discards the definition, and adds the previous version to its rules.

Note that this method of learning exceptions is quite different from simply applying CRYSTAL recursively. I will refer to this method as "single-feature" exception learning to emphasize the difference.

The basic CRYSTAL algorithm begins with the most specific definition that covers a seed instance and generalizes as far as the training allows. When the training set is sparse, the definitions it learns tend to be too tightly constrained to have good coverage on the test set.

In contrast to this, exception learning begins with highly generalized exceptions. All but a single feature of the negative instance has been dropped. This increases the likelihood that the exceptions will apply to test instances as well as training instances. This is an important consideration since exceptions are learned from a limited training sample.

```
The CRYSTAL algorithm with exceptions:
        Rules = NULL
        Derive an initial concept definition from each positive instance
        Do for each initial definition D not covered by Rules
            Loop:
                D' = the most similar definition to D
                If D' = NULL, exit loop
                U = the unification of D and D'
                Test U on the training set
                If the error rate of U > Tolerance
                    Call Add_Exceptions(U)
                    If the error rate of U > Tol2
                        Exit loop
                Set D = U
            Add D to the Rules
        Return the Rules

Add_Exceptions(U)
        Covered = instances covered by definition U
        If no more than half the covered instances are negative
            Do for each negative instance N in Covered
                Add each feature of N to list of possible exceptions
            Do for each feature F in possible exceptions
                If F is found in at least two negative instances
                    Excluded = instances in Covered with feature F
                    If percentage of positive instances in Excluded ≤ XTol
                        Add F to Exceptions of U
                        Update the error rate of U
        Return U
```

Figure 6.3 The CRYSTAL algorithm with exception learning

Where the basic CRYSTAL algorithm uses the entire training set to generate a definition, an exception must be learned in the context of the region of instance space covered by a single definition. This tends to be two or three orders of magnitude smaller than the entire instance space. In such a small training set, overfitting becomes a serious problem.

CRYSTAL does not even try to learn exceptions when a definition is so overgeneralized that it covers more negative instances than positive. In such cases it is unlikely that exceptions can bring the error rate back within tolerance, but considerable computation time will be expended.

Learning exceptions increases CRYSTAL's time complexity, particularly when e, the number of extraction errors, becomes large. Computation time of the basic CRYSTAL algorithm is proportional to n for each proposed generalization, where n is the number of training instances. Learning exceptions adds $O(e^2)$ computations to assemble the list of features found in at least two errors and $O(e^2)$ computations to test the candidate exceptions on instances covered by the definition.

This brings the amount of computation for each generalization to $O(n+e^2)$ when exceptions are being learned and $O(n)$ otherwise. The overall computation time with exceptions is $O(pn + pe^2)$, as compared to $O(pn)$ without exceptions.

For the training sets used in these experiments, e^2 was generally less than n. The actual computation time with exceptions was typically about one and a half times as long as without exceptions.

How much does this increase in computation time buy in terms of precision and recall? It depends on the characteristics of the instance space, as shown in the next section.

6.1.2 Empirical Results

In the following experiments CRYSTAL was run with and without exception learning at error tolerance 0.20. Error tolerance of definitions with exceptions (Tol2) was set at 0.10 and exception tolerance (XTol) at 0.20.

Figure 6.4 shows results from the Management Succession domain and Figure 6.5 from the Hospital Discharge domain.

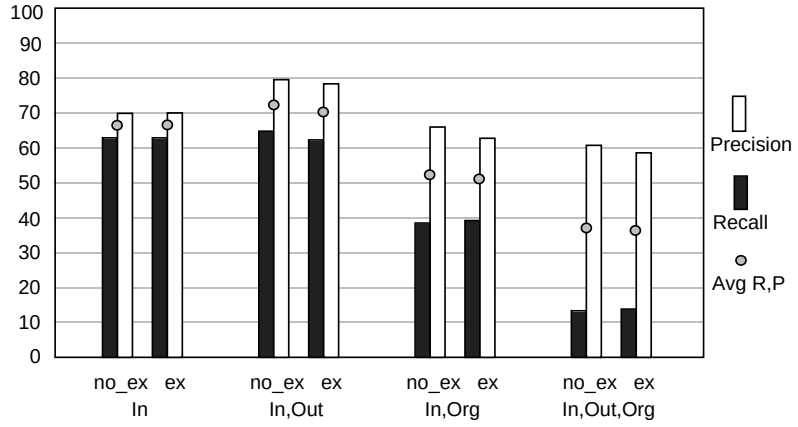

Figure 6.4 Comparison of Management Succession results with no exceptions and with exception

Learning exceptions with the parameter settings used here has little effect in the Management Succession domain, and causes a small increase of recall at the expense of precision in the Hospital Discharge domain.

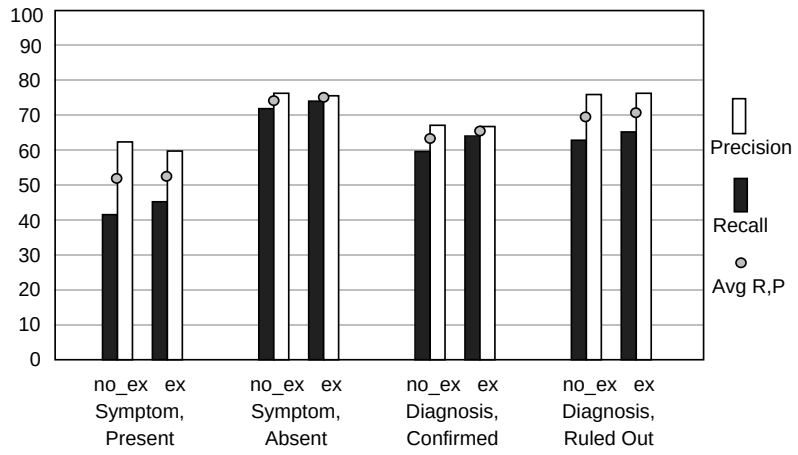

Figure 6.5 Comparison of Hospital Discharge results with no exceptions and with exception

Of the Management Succession concepts shown here, only *In,Out* has a statistically significant difference from learning exceptions[1]. Even that is only a tiny drop in performance.

Each Hospital Discharge concept gets a small, but statistically significant boost in recall from exceptions. The gain in recall is slightly more than the loss in precision. The gain in average recall and precision is significant for all but *Diagnosis,Ruled_Out*.

There is an intuitive appeal to including negative as well as positive constraints in a concept definition. It is possible that another strategy for learning exceptions would allow CRYSTAL to boost precision without sacrificing recall. The methods I have tried so far, have not succeeded in showing this to be possible.

An attempt to enhance the expressiveness of CRYSTAL's rule representation made only a tiny performance improvement in one domain and no im-

[1]The significance test used here and elsewhere is a two-tailed, paired t-test with $p<0.05$.

provement in another. The next section presents experiments that restrict CRYSTAL's representation.

6.2 Restricting CRYSTAL's Representation

CRYSTAL was designed with the assumption that a rich and flexible representation was needed to express the variability of unrestricted text. Semantic class constraints and the ability to drop all but one or two constraints allow CRYSTAL to learn highly general rules that give broad coverage on previously unseen instances.

On the other hand, CRYSTAL needs the ability to use a wide variety of evidence to find reliable rules when semantic class constraints alone are not adequate for the extraction task.

This section presents a series of experiments that test these assumptions. It is not certain that increasing the complexity of the rule representation will help performance. Overly complex rules increase the dimensionality of the search space and may tend to make learning more difficult. With more possible ways to relax constraints, CRYSTAL will have more ways to make wrong choices.

I selected four aspects of CRYSTAL's representation to be crippled. The first is CRYSTAL's ability to either keep or drop verb constraints. This ability is lacking or quite limited in other systems that learn text analysis rules (see Chapter 7). How much does CRYSTAL gain from its ability to drop verb constraints? The results labeled with a "V" are from a version of CRYSTAL that never generalizes away the verb of the seed instance.

This was accomplished by restricting the candidates for most similar definition to those with the same verb root. All generalizations from a seed instance

included a verb constraint with at least the same root form. Seed instances from sentence fragments with no verb were unified only with instances with no verb and produced definitions that require a null verb.

A second aspect is the distinction between head terms and modifiers for the term constraints and class constraints. This distinction is unique to CRYSTAL. The version labeled "M" has term constraints and class constraints, but omits the head and modifier constraints on terms and classes from the initial definitions. The generalized definitions lack these constraints as well.

A third aspect is the inclusion of all syntactic constituents as possible constraints in the concept definition. Other systems that learn text analysis rules include only certain constituents, such as the verb plus those containing extracted information. The version of CRYSTAL labeled "X" omits all constraints from initial definitions but those on the verb and on extracted constituents. For example if the seed instance has information extracted from the direct object, the initial definition will have only the verb and direct object. The subject and any prepositional phrases will be immediately dropped.

A fourth aspect is the ability to use either term constraints or semantic constraints on any syntactic constituent. Other systems allow term constraints only on a "trigger" word, typically the verb or verb root. The version labeled "T" retains the root constraint on verbs, but does not include any term constraints.

Results also include the baseline CRYSTAL system with none of these restrictions and a version labeled "4" that has all four restrictions. The number of possible features for this last version is much smaller than for the baseline system. If the features remaining are sufficient to describe the target concept,

it may be an advantage to eliminate the unnecessary features. If these restrictions have removed features that are essential to distinguish the positive instances, performance should drop drastically.

Figure 6.6 shows results for two of the Hospital Discharge concepts. These are representative of all four concepts in this domain.

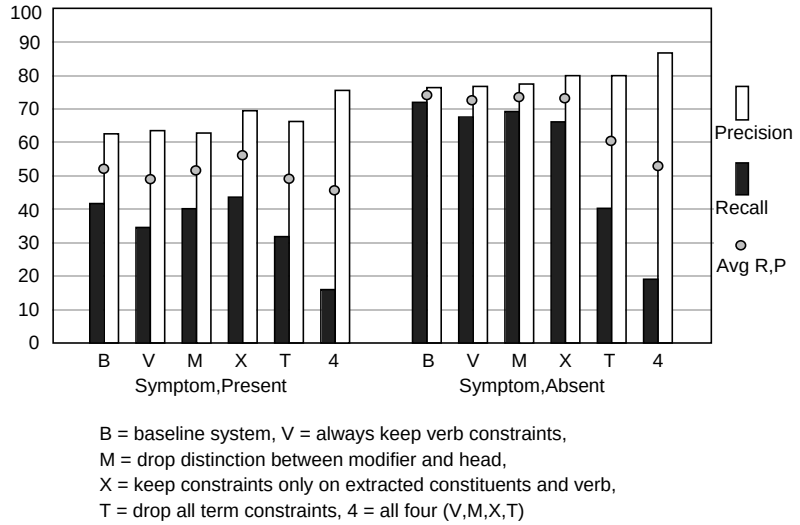

Figure 6.6 The effects of restricting CRYSTAL's representation in the Hospital Discharge domain

Single restrictions tend to cause a moderate drop in recall and a rise in precision[2]. Of the single restrictions, dropping terms hurt recall the most.

This suggests that CRYSTAL's representation offers enough alternate ways to describe the positive instances that CRYSTAL can compensate for mild crippling of its representation. When all four restrictions are applied, recall plummets. The severely crippled representation is insufficient to distinguish many of the positive instances.

[2] All the differences from the baseline system for *Symptom,Absent* are significant, except for the difference in precision for "V". Most changes for *Symptom,Present* are significant, except recall for "X", precision for "M", and average precision and recall for "M".

I had expected that the effects of restricting representation would be much less pronounced after semantic fine-tuning. Figure 6.7 shows the same pattern of results, however, as before semantic fine-tuning. The effect of dropping term constraints is less pronounced with better semantic tagging.

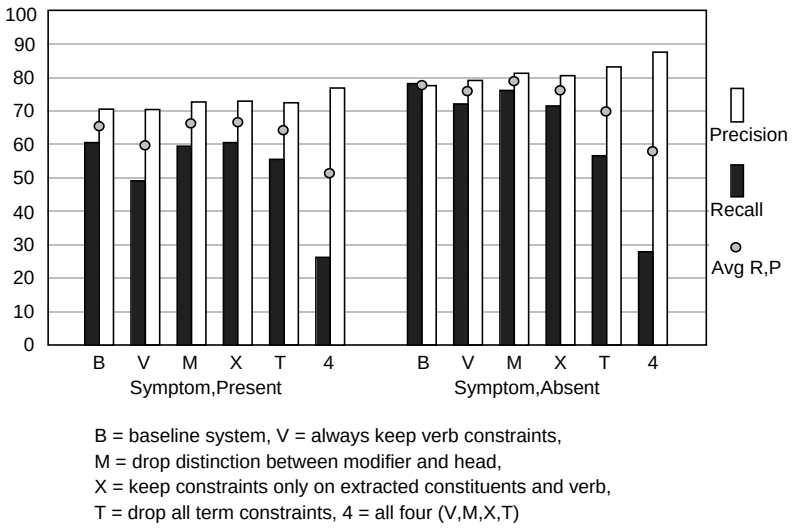

Figure 6.7 Restricting CRYSTAL's representation with fine-tuned semantics in the Hospital Discharge domain

Figure 6.8 shows results of these restrictions in two of the Management Succession concepts. The behavior of *Person_In* is characteristic of most of the Management Succession concepts. Any single restriction had a minor effect on performance. Precision tends to be a little higher and recall a little lower for each of the restrictions[3]. The average of recall and precision is only slightly different from the baseline system for any single restriction. Applying all four restrictions causes a large drop in recall.

[3]The drop in recall from the baseline system is statistically significant only for versions "X", "T" and "4". The increase in precision is significant for all but "T". The change in average recall and precision is significant only for "V", "T", and "4".

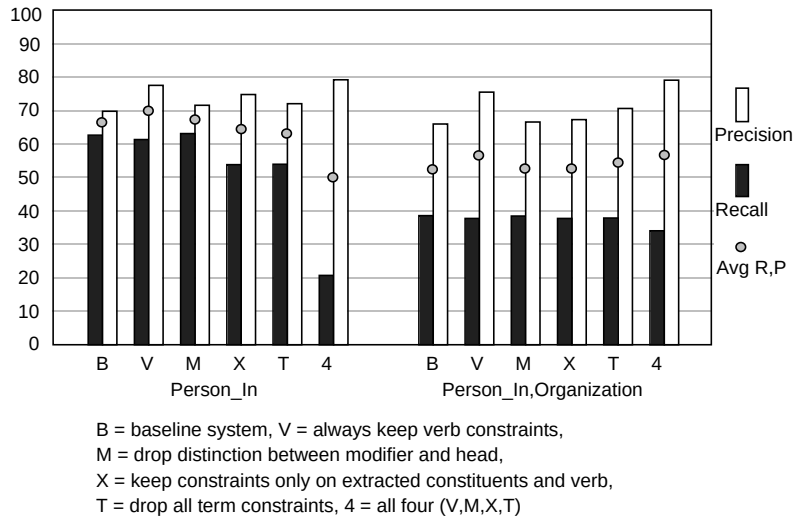

Figure 6.8 The effects of restricting CRYSTAL's representation in the Management Succession domain

For some of the multi-slot concepts, the severely restricted representation was more nearly adequate. For three of the fifteen Management Succession concepts, including *Person_In,Organization*, applying all four restrictions increased precision more than it hurt recall. No single restriction affected recall for *Person_In,Organization* and "V" and "T" raised precision somewhat[4]

6.3 Discussion of Results

The results in this chapter indicate that representation can have a major impact on performance and on the ability to learn text analysis rules. The instance representation and the rules must be expressive enough to represent the essential characteristics of the concept being learned. If this minimum

[4]The only significant change in recall from the baseline system was for version "4". The increase in precision and in the average of recall and precision was significant only for "V", "T", and "4".

expressiveness has been met, adding unnecessary features will only hinder a machine learning algorithm.

For the concept *Person_In,Organization* in the Management Succession domain, a simple rule representation was sufficient. Adding exceptions to the rules only lowered precision with no gain in recall. Each of the restrictions to CRYSTAL either had no impact on this concept or raised precision with no change to recall. Applying all the restrictions lowered recall slightly, but produced a gain in precision.

One way to look at this behavior is in terms of the nature of the instance space for *Person_In,Organization*. Simple rules are sufficient to describe one third of the positive instances reliably. The remainder are the "hard" instances (as discussed in Section 5.4.2). Increasing the expressive power of CRYSTAL has two conflicting effects. It allows rules that identify more of the hard instances, but at the same time increases the potential for CRYSTAL to relax the wrong constraint while generalizing a definition.

For most of the other other concepts in these two domains, a simple representation is inadequate to identify more than a small fraction of the positive instances. An example of this is the concept *Symptom,Present* in the Hospital Discharge domain. This was the most difficult concept of either domain, particularly before semantic fine-tuning. Most of the positive instances were "hard" instances. CRYSTAL needed all its expressiveness including exceptions to gain recall for this concept. Nearly all the restrictions to CRYSTAL lowered the average recall and precision.

The only aspect of CRYSTAL's representation that did not help for this concept was constraints on non-extracted constituents. The verb and the

phrase containing the *Symptom,Present* are generally enough to identify a reference to this concept.

No aspect of CRYSTAL's representation is either essential for all concepts in all domains or unnecessary for all concepts. Any restriction that hurt performance for one concept turn out to help another concept. Each of these possible restrictions have been left as options for the CRYSTAL system, with the default being the full representation.

CHAPTER 7

RELATED WORK IN NATURAL LANGUAGE PROCESSING

Most research in applying machine learning to natural language processing has been primarily at the level of lexical or semantic disambiguation of individual words [Brill 1994, Cardie 1993, Yarowsky 1992, Church 1988] and in learning heuristics to guide probabilistic parsing [Charniak 1995, Magerman 1995]. Little work has been done, however, in using corpus-based techniques for a higher level of inferencing that goes beyond the meaning of individual words.

The work most closely related to CRYSTAL has come from participants in recent Message Understanding Conferences [MUC-4 1992, MUC-5 1993, MUC-6 1995]. Nearly all MUC participants use some form of pattern matching rules or finite state automata to identify references to concepts of interest.

Although most MUC participants build these rules by hand, the methodology looks uncannily like a hand-simulation of CRYSTAL. Actually it is CRYSTAL that automates the iterative development cycle a human uses. A knowledge engineer applies an existing rule base to a set of annotated training texts, selects a phrase not covered by the rules, and adds a new rule or generalizes an existing rule to cover it. The modified rules are tested on the training set to ensure that new rules do not create excessive errors.

Both the human developer and CRYSTAL rely heavily on examples from a representative set of texts. The human has the advantage of outside knowl-

edge that helps predict which features are important to include in a rule and how far a rule may be safely generalized. The quality of semantic tagging of individual words and the expressiveness of the rules themselves play a critical role, whether the rules are learned or hand-crafted.

Three of the MUC participants have developed systems that learn text analysis rules. The remainder of this section will compare CRYSTAL with these other systems, plus another system based on a MUC domain. The first is the AutoSlog dictionary construction tool [Riloff 1993] used by the University of Massachusetts in MUC-4 and MUC-5. AutoSlog combines machine learning with a "human in the loop" who edits the proposed rules.

The second system is PALKA [Kim and Moldovan 1992], developed by the University of Southern California for their MUC-5 system. A third trainable system appeared in MUC-6, the HASTEN system from SRA [Krupka 1995]. The fourth system described in this chapter is LIEP [Huffman 1996] that was developed on the MUC-6 domain. By the time of the MUC-6 conference, the University of Massachusetts had moved from AutoSlog to CRYSTAL.

7.1 AutoSlog

The AutoSlog dictionary construction tool was developed by Ellen Riloff at the University of Massachusetts [Riloff 1993]. AutoSlog passes through the training texts a single time and proposes concept definitions from instances of the concepts to be extracted.

AutoSlog uses "one shot" learning with no generalization phase and no testing of proposed rules on the training data. Instead, it uses heuristics to craft the best concept definition it can from a single motivating example. An

AutoSlog concept definition assigns a fixed level of semantic constraint to the extracted phrase. A more recent version of AutoSlog [Riloff 1996] has no semantic constraints at all on the extracted phrase. This version assumes that later processing in an information extraction system will filter out extraction errors by overgeneralized AutoSlog concept definitions.

Each AutoSlog definition also has an exact word constraint on a *trigger* word, which is determined by a set of rules. When extracting a concept from the subject, AutoSlog selects the verb as trigger, in some cases including the direct object or infinitive complement as well. Extraction from a direct object is likewise triggered by the verb. Extraction from a prepositional phrase may be triggered either by the preceding noun or by the preceding verb.

One major difference between CRYSTAL and AutoSlog is AutoSlog's relatively limited representation. The phrase being extracted must always have a fixed level semantic constraint, specified in advance by the user, and never has an exact word constraint. No other phrase in the instance may have a semantic constraint. Every concept definition must have an exact word constraint on a trigger word. No other phrases may be included in the definition.

Another difference is AutoSlog's lack of a mechanism for automatically testing its proposed concept definitions on the training corpus. AutoSlog relies on human review before the concept definitions are finally accepted. This human review requires only a few hours and typically retains about 30% of the proposed concept definitions as reasonable.

AutoSlog does remarkably well despite its limitations, when supported by carefully engineered semantic tagging and subsequent discourse processing that filters out extracted phrases that are irrelevant to the extraction task. An Au-

toSlog dictionary achieved 98% of the performance of a hand-crafted concept dictionary that took an estimated 1500 hours of effort to create for the MUC-3 evaluation.

The system presented in the next section uses a covering algorithm approach that resembles CRYSTAL more closely than AutoSlog does.

7.2 PALKA

The PALKA system [Kim and Moldovan 1992] was developed by Jun-Tae Kim and Dan Moldovan for the University of Southern California MUC-5 system. PALKA (Parallel Automatic Linguistic Knowledge Acquisition) uses a method similar to the candidate elimination algorithm (See Section 8.3.4) to generate text analysis rules from training instances. As PALKA considers each new instance, it generalizes rules to include positive instances not yet covered and specializes rules to avoid covering negative instances.

PALKA represents its rules as *FP-structures* (Frame-Phrasal pattern structures), which have a constraint on the verb root and semantic constraints on each extracted phrase. An FP-structure may be generalized by moving a semantic constraint upwards in a semantic hierarchy or by adding a disjunctive term to the semantic constraint.

If an FP-structure has a constraint on class A, but a negative instance has class A' (a subclass of A), PALKA will specialize by moving lower in the semantic hierarchy and enumerating all classes except A'. PALKA lacks CRYSTAL's error tolerance mechanism and will specialize a FP-pattern based on a single negative instance even if this excludes several positive instances as well.

PALKA maintains a single rule for each phrasal pattern, which may have several disjunctive terms in each semantic class constraint. Here is an example of a possible FP-pattern for a *Bombing* case frame, with *Instrument* and *Target* slots. The subject must have class *dynamite* or *grenade*; the verb must have roots "be" and "hurl"; the object of the preposition "at" must be of class *physical_object*. This would cover an instance such as "Dynamite sticks were hurled at U.S. Embassy facilities."

Bombing
(Instrument: dynamite ∨ grenade)
BE HURL
AT
(Target: physical_object)

Like AutoSlog, PALKA has a more restrictive representation than CRYSTAL. Each FP-structure requires a constraint on the root form of a verb, but can have no other exact word constraints. The subject, direct object, and each extracted phrase have semantic constraints, but PALKA does not allow constraints on prepositional phrases other than those containing information to extract.

In some ways, PALKA's representation is even more limited than AutoSlog's. FP-structures cannot express exact word constraints on a noun instead of a verb, and hence cannot represent the rule: "ATTACK ON (Target: physical_object)" where "attack" is a noun.

7.3 HASTEN

The next trainable text analysis system, HASTEN, uses a form of instance based learning similar to PEBLS, which will be introduced in Section 8.4. George Krupka developed HASTEN [Krupka 1995] for the SRA Corporation's MUC-6 text analysis system.

HASTEN stores each training instances as an *Egraph*, which associates structural elements of the sentence with semantic classes and also with case frame slots of an extracted concept. A new instance is classified by computing its similarity to each stored Egraph. If the similarity to the nearest Egraph is above a threshold, HASTEN extracts information from the new instance based on the Egraph case frame.

The following example is an Egraph for the input "Armco also named John C. Haley, 64 years old, chairman." This example contains a Management Succession event with *Organization*, *Person_In*, and *Position*. An Egraph also has an *anchor*, which is the main element of the instance, generally a verb phrase. This is similar to the trigger in AutoSlog.

Succession

Organization:	NP sem=not-govt
Anchor:	VP root=name
Person_In:	NP sem=person
(irrelevant):	AGE
Position:	LIST sem=post

HASTEN uses a similarity metric to find the most similar Egraph to a new input sentence. This metric considers how many structural elements match, how well the semantic contents match, and whether relative ordering and

adjacency of elements are maintained. A number of tunable parameters are used in the similarity metric.

Some Egraphs will do a more reliable job of identifying concepts than others. HASTEN evaluates the classification performance of each Egraph on the other training instances and assigns it an *extraction bias*. HASTEN multiplies the similarity measure by this extraction bias to reduce the effective similarity to Egraphs with poor classification performance.

HASTEN also has a user-defined threshold that manipulates a trade-off between recall and precision just as CRYSTAL's error tolerance does. If the computed similarity between an input sentence and the most similar Egraph falls below the threshold, nothing is extracted. A high threshold makes HASTEN more cautious and increases precision, while a low threshold increases recall.

CRYSTAL's representation of rules is in some ways more flexible than HASTEN's representation. CRYSTAL can set constraints on certain structural elements of a sentence and not others, depending on the particular concept definition. HASTEN's similarity metric uses the same weight for a given structural element for all Egraphs.

If HASTEN's similarity metric gives a high weight to the anchor, then in effect HASTEN cannot drop constraints on the verb from its implicit rules. CRYSTAL rules can also be selective about including semantic class constraints on some elements and exact word constraints on others, but HASTEN has no such flexibility.

The examples of Egraphs in the published account had only semantic constraints and not word constraints on the sentence elements used as slot fillers.

The Egraph anchor had a constraint on the verb root, and the irrelevant sentence element had no constraints.

7.4 LIEP

The last system to be discussed in this chapter is Scott Huffman's LIEP (Learning Information Extraction Patterns) [Huffman 1996]. LIEP uses a set of heuristics to create rules, called *extraction patterns*, from single training instances. There is also a mechanism to generalize extraction patterns slightly.

LIEP learns patterns for multi-slot concept extraction, such as Management Succession events. Unlike AutoSlog's heuristics that operate on single slots, LIEP's heuristics operate only on multiple slots. In many ways, LIEP functions as a multi-slot version of AutoSlog.

A key word filter is applied to input sentences before presenting instances to LIEP's extraction rules. This gives LIEP a training corpus that consists almost entirely of positive instances, although not all phrases of the instances will be relevant and particular extraction patterns will apply to only a fraction of the entire training.

Extraction rules have syntactic constraints on pair-wise syntactic relationships between sentence elements. LIEP finds relationships that link the extracted phrases in an instance, and includes non-extracted sentence elements only if needed to form a path between extracted elements. These additional sentence elements are typically verbs and prepositions.

For example, a pattern from the instance "Bob was named CEO of Foo Inc." has the following syntactic constraints. The verb "named" and the

preposition "of" are included to link the extracted constituents "Bob", "CEO", and "Foo Inc.".

```
subject(Bob,named)
object(named,CEO)
post_nominal_prep(CEO,of)
prep_object(of,Foo Inc)
```

Extraction patterns also have semantic constraints on the extracted phrases. LIEP uses whatever semantic class it finds in the motivating instance and does not generalize the semantic constraints. Non-extracted elements have exact word constraints, and in the case of verbs a constraint on active or passive voice.

LIEP proposes up to three rules from each example and then tests each one on the training set. Of the alternatives proposed, LIEP keeps the best one according to a metric that combines recall and precision[1].

A limited amount of generalization of extraction patterns is done. When two patterns are identical except for word constraints, LIEP builds a synonym list from those words. For example, "named", "appointed", and "elected" are synonyms in Management Succession patterns. Any pattern that includes a word in a synonym list has the word constraint generalized to a constraint on the synonym list.

This will not help LIEP generalize to words not found in training. Huffman suggested use of a knowledge source such as WordNet [Miller *et al.* 1990] to replace or augment the learned synonym lists.

[1]The "F-measure" used in MUC evaluations to combine recall and precision

There does not seem to be a mechanism to discard unreliable extraction patterns. Perhaps the key word filter eliminates enough of the negative instances before they are presented to LIEP that overgeneralized extraction patterns are not a major problem.

7.5 Contrast of CRYSTAL and Other NLP Algorithms

The four systems described in this chapter have a wide range of strategies for learning text analysis rules. At one extreme, AutoSlog and LIEP rely on heuristics to derive an extraction rule from a single positive instance. At the other extreme, PALKA uses a computationally expensive learning method that generalizes a rule to include other positive instances and specializes to avoid negative instances. HASTEN takes an instance based learning approach and bases its classification on comparing a new instance to stored training instances. What HASTEN learns is weights that indicate the classification accuracy of each stored instance.

One thing in common to all four is a much more limited rule representation than CRYSTAL's. CRYSTAL's concept definitions may have either term constraints or semantic constraints on any syntactic constituent in the instance. No distinction is made in advance between sentence elements that contain extracted information and those that do not.

The other systems allow only semantic constraints on extracted sentence elements. Only certain non-extracted elements (e.g. the verb, "trigger", or "anchor") are included, and these have term constraints, but not semantic constraints.

These restrictions in representation can be viewed as a strong bias in the learning algorithms. Very little search is needed for an appropriate rule if certain decisions are made in advance. In the case of AutoSlog the restrictions are so strong that no search is done at all, and hardly any search in the case of LIEP.

HASTEN uses the same distance function for all its stored examples, rather than learning what features of an instance are essential for classification. The essential features are decided in advance by the way HASTEN represents its stored instances.

PALKA does extensive search, but with a learning algorithm that cannot navigate large search spaces efficiently. PALKA restricts the search space to a manageable size by considering only semantic constraints for extracted sentence elements.

CRYSTAL's expressive representation is made possible by its efficient search strategy. CRYSTAL can include any feature in its representation that might possibly be useful. Features that are irrelevant to the target concept are quickly dropped when generalizing a concept definition.

In most cases, the essential features turn out to be those included in the representation of the other systems. The semantic class of extracted phrases together with the verb root is often sufficient context for a text analysis rule. However, as experiments in Section 6.2 show, this is often not enough. Including a wide array of features in CRYSTAL's representation allows CRYSTAL to learn rules that could not be otherwise expressed.

CHAPTER 8

RELATED WORK IN MACHINE LEARNING

This chapter compares CRYSTAL with related machine learning algorithms. Since the other algorithms discussed here are *classifiers* that assign a class such as positive or negative to each instance, I begin by discussing how CRYSTAL could be transformed into a classifier. I then discuss the extremely large number of features that arise from CRYSTAL's rule representation and how these features could be expressed in the predicate calculus or as the feature vectors that some algorithms expect. One point of comparison with other algorithms is how well they handle large feature spaces.

The first class of related machine learning algorithms I discuss are covering algorithms, the family to which CRYSTAL belongs. Next, CRYSTAL's use of a distance function is contrasted to instance based learning, or "k-nearest neighbor" algorithms. Lastly, CRYSTAL is compared to the well-known family of decision tree and decision list algorithms. The chapter concludes with some observations on the contrast between CRYSTAL and these other machine learning algorithms.

8.1 CRYSTAL as a Machine Learning Classifier

Among machine learning algorithms, classifiers are those that take each instance and assign to it one of k disjoint classes. A broad range of problems

can be formulated in terms of classification. An instance might represent a situation and the classes might be several alternate actions that could be taken, depending on the situation.

Information extraction can be turned into a classification problem. A set of concept definitions classify an instance as positive if the target concept is found in the instance and negative, otherwise. The evaluation procedure I use to compute the recall and precision of a set of concept definitions treats CRYSTAL as such a classifier. What percentage of the positive instances does CRYSTAL classify correctly? Of the instances that CRYSTAL classifies as positive, what percentage are true positives?

Text analysis rules must go further, however, than simply to decide whether an instance contains the target concept. The rules must also indicate where to extract information from the instance. In terms of CRYSTAL's concept definitions, this means specifying a mapping between slots in the target concept and syntactic constituents of the instance. This is comparable to a binding problem in a predicate calculus representation.

Consider the sentence in Figure 8.1 from the Hospital discharge domain. It has been annotated with a *Symptom,Absent* ("regular rate") in one prepositional phrase and a *Symptom,Present* ("early beats") in another prepositional phrase. A concept definition for *Symptom,Absent* must specify that the target information is in the first prepositional phrase, while a concept definition for *Symptom,Present* must specify the second prepositional phrase.

Note that the concept definitions for a domain cannot be viewed as a single classifier with a class for each possible concept, plus the classification of "none" for irrelevant sentences. That would not work, since a sentence may

Input Sentence:
The heart was notable for a <SA> regular rate </SA>
with occasional <SP> early beats </SP>.

CRYSTAL Instance:
SUBJ:
Terms: THE HEART
Classes: <Body Part>
Mode: affirmative
VERB:
Terms: WAS
Root: BE
Classes: <Exist>
Mode: active, affirmative
PP:
prep: NOTABLE_FOR
Terms: A REGULAR RATE
Classes: <Temporal Concept>
Mode: affirmative
Extract: Symptom,Absent
PP:
prep: WITH
Terms: OCCASIONAL EARLY BEATS
Classes: <Temporal Concept>
Mode: affirmative
Extract: Symptom,Present

Figure 8.1 An annotated instance with *Symptom,Absent* in one prepositional phrase and *Symptom,Present* in another.

contain references to multiple concepts or more than one reference to the same concept. The instance in Figure 8.1 cannot be classified as *Symptom,Absent* and as *Symptom,Present* simultaneously.

Would it be sufficient for each concept in the domain to have its own classifier that takes one of CRYSTAL's instances and classifies it as positive or negative? That would not be enough to specify *where* the target concept was to be found in the instance. Even a classifier with a separate class for each

possible syntactic constituent would not do the job. If a *Symptom,Absent* classifier returned the class of "PP" for the instance in Figure 8.1, this would still not be specific enough to find "a regular rate" in the instance.

Creating a classifier equivalent to a CRYSTAL concept definition is a bit tricky. A single instance for CRYSTAL must be presented as multiple instances, each one designating a particular mapping of syntactic constituent and concept slots. If the concept has s slots and the instance has k syntactic constituents, the CRYSTAL instance would be transformed into sk separate instances for the classifier.

The concept *Symptom,Absent* has one slot and the instance in Figure 8.1 has four syntactic constituents, so a *Symptom,Absent* classifier is presented with four instances. Figure 8.2 shows the four mappings and their classification. A *Symptom,Present* classifier is presented with the same four instances, but returns different classifications.

Symptom,Absent classifier :	
Symptom,Absent in Subj?	negative
Symptom,Absent in Verb?	negative
Symptom,Absent in First PP?	positive
Symptom,Absent in Second PP?	negative
Symptom,Present classifier:	
Symptom,Present in Subj?	negative
Symptom,Present in Verb?	negative
Symptom,Present in First PP?	negative
Symptom,Present in Second PP?	positive

Figure 8.2 Mapping of concept slots to syntactic constituents.

Multi-slot concepts, such as those in the Management Succession domain require a separate instance for every mapping of concept slots and syntactic

constituents in the instance. A CRYSTAL instance with four syntactic constituents would be transformed into twelve instances for a classifier for the three slot concept with *Person_In*, *Position*, and *Organization*.

Consider the sentence "XYZ Inc. named Mr. A as president, replacing Mr. B". This sentence has five syntactic constituents (subject, verb, direct object, prepositional phrase, and relative clause) and fifteen possible mappings of the three slots to these five constituents. The only positive instance is the one that designates that *Organization* is in the subject, *Person_In* is in the direct object, and *Position* is in the prepositional phrase.

Now that CRYSTAL concept definitions have been shown to be functionally equivalent to a classifier, it may be compared to other machine learning classifiers. The only thing to bear in mind is that if CRYSTAL has n instances, a classifier will need skn instances, where s is the number of concept slots and k is the average number of syntactic constituents in an instance. As long as both s and k are small, this may be treated as a constant factor in the amount of computation time needed for machine learning classifiers.

8.2 Extremely Large Feature Sets for Natural Language Processing

In CRYSTAL's representation language, each instance has only a small number of features, but thousands of features may be needed to express constraints on all the distinct terms and semantic classes found in a training corpus. This is considerably larger than the typical number of features for experiments reported in the machine learning literature.

Some of the data sets often used in published results are from the UC Irvine repository of machine learning data sets. A data set on predicting recurrence of breast cancer has instances with nine features, of which four take integer values. A data set on recognizing digits in a faulty LCD display has seven Boolean features, one for each of the lines that form the displayed digit. A data set on determining the winner of chess end games has instances that use 39 Boolean features to represent board positions.

Published accounts of the algorithms that I discuss in this chapter report experiments on data sets that have a fairly small number of features. A data set on diagnosing soybean diseases was used by Michalski to test the A^q algorithm. Each instances has 35 features with a total of 106 feature-value pairs [Michalski 1983]. Clark and Niblett report tests of CN2 on three medical data sets with eighteen, nine, and seventeen features and two artificial domains with twelve features [Clark and Niblett 1989]. Mitchell used mass spectroscopy data to test the candidate elimination algorithm. Each instance represents an molecule with four features for each of ten to twenty atoms in the molecule [Mitchell 1978]. Pagallo and Haussler tested GREEDY3 on artificial data sets with instances created by randomly selecting from among sixteen Boolean features [Pagallo and Haussler 1990] in some tests and from eighty in others.

I have not seen any machine learning papers that describe a data set with more than several dozen features. There often seems to be an implicit assumption that the number of features will be fairly small, and certainly much smaller than the number of instances. This is not the case when text analysis rules are learned using CRYSTAL's representation.

My motivation for creating CRYSTAL was to have a learning algorithm that could handle extremely large numbers of features easily. I had run into the problem of extremely large number of features in an earlier system called WRAP-UP [Soderland and Lehnert 1994] that learns to make inferences during the discourse processing stage of information extraction. Features are created that represent particular words in particular syntactic roles, such as being the subject of a particular verb, the object of a particular verb, or the object of a particular preposition. This results in over a thousand features, even after discarding features that are not found in at least ten texts. Inducing decision trees[1] with such a large number of features took an excessive amount of memory and computation time.

An extremely large number of features also arises for CRYSTAL in learning text analysis rules. A corpus of texts will typically include thousands of distinct words that can occur in a variety of syntactic roles, resulting in tens of thousands of combinations of word and syntactic role. Figure 8.3 shows the features occurring in a training instance with "Chest x-ray" in the subject.

SUBJ-Terms-CHEST
SUBJ-Terms-X-RAY
SUBJ-Mod_Terms-CHEST
SUBJ-Head_Terms-X-RAY
SUBJ-Classes-Body_Location
SUBJ-Classes-Diagnostic_Procedure
SUBJ-Mod_Classes-Body_Location
SUBJ-Head_Classes-Diagnostic_Procedure
SUBJ-Mode-affirmative

Figure 8.3 Boolean features derived from "Chest x-ray" in the subject.

[1] I used a slightly re-implemented ID3 decision tree algorithm [Quinlan 1986]

These features are listed in a format that indicates the syntactic constituent (SUBJ), followed by the constraint (e.g. Terms), followed by the word or semantic class. The feature SUBJ-Terms-CHEST has the value "true" for this instance as does the feature SUBJ-Terms-X-RAY.

The constraints on an instance can also be represented in the predicate calculus. One encoding scheme is to treat constraints as predicates with two arguments. The predicate name is the constraint (e.g. Terms), the first argument is the syntactic constituent, and the second is the word or semantic class. The term "chest" in the subject is expressed as Terms(SUBJ, CHEST) and the term "x-ray" as Terms(SUBJ, X-RAY). This is shown in Figure 8.4.

Terms(SUBJ, CHEST)
Terms(SUBJ, X-RAY)
Mod_Terms(SUBJ, CHEST)
Head_Terms(SUBJ, X-RAY)
Classes(SUBJ, Body_Location)
Classes(SUBJ, Diagnostic_Procedure)
Mod_Classes(SUBJ, Body_Location)
Head_Classes(SUBJ, Diagnostic_Procedure)
Mode(SUBJ, affirmative)

Figure 8.4 Predicate calculus representation of "Chest x-ray"

Can the constraints in CRYSTAL's representation can be converted into features that take multiple values? If a feature took as its value any of thousands of terms, only a small number of features would be needed, although the number of feature-value pairs would not be changed. Such an encoding scheme will not work, however, since a given syntactic constituent may have multiple terms, multiple classes, and so forth. If the value of SUBJ-Terms is "CHEST", the same feature cannot simultaneously have the value "X-RAY".

The list of possible syntactic constituents will vary according to the syntactic analyzer used to create input for CRYSTAL. In the Hospital Discharge domain, there are five syntactic constituents used: subject, verb, direct object, indirect object, and prepositional phrase. When term constraints, head term constraints, and modifier term constraints are taken into account, a constraint may specify any of fifteen syntactic roles for a given word. Some of these combinations will not actually occur in the training data for a particular word or class, and can be ignored.

The corpus of 502 Hospital Discharge documents contains 10,710 distinct words, which results in 54,902 features, where each distinct combination of a word or semantic class occurring in a syntactic role is considered to be a feature[2]. This number can be reduced somewhat by discarding low frequency features, although this runs the risk of discarding some potentially useful features. After discarding those not found in at least ten documents, 4,697 features remain.

Expressing an instance as a Boolean feature vector requires space proportional to the total number of features found in the training corpus. Only a tiny fraction of these features will be true for a given instance. CRYSTAL's representation or a predicate calculus representation takes space proportional to the features that are true for each instance and is independent of the total number of features.

With such a large number of features, it is desirable to use an algorithm with computation time that does not depend on the total number of features. In the following sections, I pay close attention to the computational complexity

[2]This was computed by creating instances for a classifier that determine whether the subject contains the concept *Symptom,Present*.

of various algorithms, in particular, how well the algorithm would scale up to thousands of features.

8.3 Covering Algorithms

CRYSTAL falls into a class of machine learning algorithms known as covering algorithms. Covering algorithms build a set of concept descriptions that cover the positive training instances while avoiding negative instances. Concept descriptions may either be generalized to include additional positive instances or specialized to exclude negative instances. As each new concept description is generated, the instances covered by the new description are eliminated from consideration in generating further concept descriptions.

This general methodology is not new, and goes back at least to John Stuart Mill in 1843. Each covering algorithm has a different strategy for making the problem computationally feasible. The covering algorithms described here are Michalski's A^q algorithm, Clark and Niblett's CN2. I include Vere's concept induction algorithm and Mitchell's candidate elimination algorithm in this section as well.

8.3.1 A^q

Ryszard Michalski and his students have published several versions of the A^q covering algorithm [Michalski 1983] and the INDUCE inductive learning program. Peter Clark and Tim Niblett also offer a readable explanation of A^q [Clark and Niblett 1989].

The basic methodology of A^q has much in common with CRYSTAL. A^q begins with a set of labeled training instances and builds a disjunctive set

of concept descriptions, which taken together cover all the positive instances and none of the negative. This set of concept descriptions, called the *cover*, is analogous to CRYSTAL's set of concept definitions. Like CRYSTAL, each step of A^q selects a positive instance not yet covered and derives a general concept description from this seed.

A^q generates this description by first building a *star* of all maximally general descriptions that cover the seed and do not cover any negative instances. A^q appears to have an implicit error tolerance of 0.0, although the algorithm could be modified to be more noise tolerant.

A^q selects the best description from the star according to a goodness metric that favors high-coverage, compact descriptions. This description is added to the cover and A^q continues its induction with a new seed instance.

Since actually finding all such maximally general descriptions would be computationally prohibitive, A^q approximates this with a beam search. A star begins with single-attribute descriptions that cover the seed. A^q maintains a set of the best s concept descriptions, those that cover the most positive and fewest negative instances. At each stage in the search, A^q considers specializations that add a single attribute to each of the s best descriptions. The door is left open as well for domain-specific heuristics that create new attributes.

Each proposed description is tested on the training set to determine the number of positive and negative instances covered. The best s of these new descriptions are retained in the star. A number of parameters is used to guide the generation of a star, including one that halts generation when a desired number of descriptions have been found that cover no negative instances.

Building a star is computationally expensive, particularly when the feature set is extremely large. The published descriptions of A^q seem to assume an implementation in which each step of specializing a concept description considers $O(a)$ specializations, where a is the number of attributes in the feature set. Each of these specializations is then tested on all n training instances. This is done for each of the s concept definitions in the star. The average number of steps in building a star is bounded by k, the maximum number of attributes per description.

Treating k as a constant, this gives $O(asn)$ computations to build each star. Let r be the number of stars built. Ideally, r depends only on the underlying concept and is independent of n. An upper bound for r is $O(p)$, where p is the number of positive training instances. (Note the parallel between k and r in the analysis of CRYSTAL's time complexity in Section 4.6.)

This gives A^q a total computational time of $O(asrn)$ or $O(aspn)$, as opposed to CRYSTAL's time complexity of $O(rn)$ or $O(pn)$. The dependency on the star size s, makes A^q resemble the beam search version of CRYSTAL. It seems that A^q performs best with a fairly large star size. Clark and Niblett report using $s = 15$ for experiments with medical data sets[3] [Clark and Niblett 1989].

So long as A^q is only tested on features sets with a few dozen attributes, it does not matter that computation time is proportional to a. For data sets in which a is extremely large, A^q would need to be re-implemented to avoid considering all possible attributes for specializations of a star.

[3]Lymphomography, breast cancer, and primary tumor data sets from the University Medical Center in Ljubljana, Yugoslavia

Suppose that A^q is used to learn text analysis rules using a representation similar to CRYSTAL's. Only a small fraction of the entire feature set could possibly be selected for a concept description covering a seed instance. There is no point in considering features for words or semantic classes[4] that do not occur in the instance. These features cannot cover the seed instance.

Assume that a seed instance has an average of t terms. A^q would need to examine each of the constraints that are based on those t terms when specializing a concept definition. This takes $O(t)$ computations. The average number of semantic classes in a sentence is bounded by the average number of terms, so $O(t)$ computations will take care of both term constraints and semantic constraints. Constraints based on terms or classes not found in the seed instance can be ignored.

If t is treated as a constant, A^q would take $O(sn)$ computations to build a star using a representation similar to CRYSTAL's. This would give A^q a time complexity that does not depend on a, similar to a beam search version of CRYSTAL.

8.3.2 CN2

CN2 [Clark and Niblett 1989], developed by Peter Clark and Tim Niblett, combines aspects of A^q with those of decision trees and decision lists. Each step of the CN2 algorithm adds a new concept description to a list of rules and then removes instances covered by the description from the training set. This is repeated until all instances are covered. Unlike A^q and CRYSTAL, which build an unordered set of concept descriptions, CN2 builds a decision

[4] or ancestors of classes found in the instance, according to a semantic hierarchy

list [Rivest 1987] of rules that are applied in order. This handles exceptions naturally. An earlier rule can remove instances that would otherwise be errors to a later rule.

Like A^q, CN2 builds concept descriptions in a top down fashion, successively adding attributes to specialize a description. CN2 does not begin from a seed instance, but starts by considering all possible descriptions with a single attribute. CN2 uses a beam search similar to that of A^q and successively specializes the best s concept descriptions.

The metric CN2 uses to select the best descriptions is the information-theoretic measure, Shannon entropy. Entropy is defined as follows, where $(p_1, ...p_k)$ is a probability distribution among k classes for the instances covered by a concept description.

$$Entropy = -\sum_i p_i log_2(p_i)$$

Entropy forms the basis of metrics used in many decision tree algorithms, as well. Minimizing entropy favors adding an attribute that comes closest to perfectly partitioning the instances into blocks of a single class. CN2 uses dynamic pruning to halt specialization of a concept description when no additional attribute makes a statistically significantly reduction in entropy.

Building a star in CN2 requires considering each of $O(a)$ attributes as the next attribute to add when specializing a star, then testing each specialization on all n training instances. This is done for each of the s concept descriptions in the star. If we treat the average number of attributes added to each description as a constant, each CN2 rule requires $O(asn)$ computations.

Unlike A^q, CN2 cannot avoid examining attributes exhaustively, since it does not use a seed instance. Any re-implementation that considers only features found in a seed would be a fundamental change to CN2. The entire decision list requires $O(asrn)$ computations, where r is the number of rules. The number of rules is bound by n, giving a worst-case time complexity of $O(asn^2)$.

8.3.3 Vere's Induction Algorithm

Steven Vere [Vere 1975] created a concept induction algorithms based on a predicate calculus representation. His main emphasis was on creating a provably correct algorithm that is guaranteed to find *all* concept descriptions consistent with the training data.

The basic step of Vere's algorithm unifies pairs of positive instances, finding the set of *maximal, consistent unifying generalizations* (mcg's). These are the most specific concept description that covers both instances and do not cover any negative instances. A later extension to the algorithm [Vere 1980] allows an mcg to include a list of exceptions, called *counterfactuals* that in turn may have counterfactuals.

The unification is actually performed on a pair of *product graphs* that represent instances or mcg's. The resulting subgraph isomophism problem is more efficient than direct manipulation in predicate calculus. The operation of computing all possible maximal unifying generalizations is still computationally expensive, however. Vere was more concerned with completeness and correctness than with computational efficiency.

A pair of positive instances will generally produce multiple mcg's. Each of these is further generalized by unifying with additional positive instances. Vere gives no time complexity analysis, but acknowledges in the 1980 paper that his approach is only "potentially analytically tractable".

8.3.4 The Candidate Elimination Algorithm

Tom Mitchell's candidate elimination algorithm [Mitchell 1978, Mitchell 1982] also identifies *all* concept descriptions that are consistent with the training data. The candidate elimination algorithm takes advantage of a partial ordering of concept descriptions to avoid enumerating the consistent concept definitions.

If a concept description s covers all the positive training instances, then so will any description d that is a generalization of s. The candidate elimination algorithm maintains a set S of the most specific concept descriptions that cover all the positive instances without covering any negative instances.

Similarly, if a concept description g avoids covering any negative training instances, then so will any description d that is a specialization of g. A second set G is maintained of the most general concept descriptions that cover all the positive and none of the negative instances.

These two sets, S and G, define the boundary of all concept descriptions that are consistent with the training. Only one pass is made through the training instances, updating S and G after each instance is read.

Mitchell analyzes the time complexity as $O(sg(p+n)+s^2p+g^2n)$, where s is the length of S, g is the length of G, and p and n are the number of positive and negative training instances. Care must be taken to avoid an explosion

in the length of S or G. Mitchell suggests alternating positive and negative training instances to help inhibit growth in the version space boundary.

The basic candidate elimination algorithm works only with perfectly consistent data. An inconsistent training instance will cause the version space of consistent descriptions to collapse to the empty set. This is a serious drawback when dealing with naturally occurring data: some instances may have incorrect feature values, and the features themselves may be inadequate to distinguish different classes of instances.

Mitchell's solution to the problem of inconsistency is to maintain multiple copies of the version space, each one consistent with as many training instances as possible. There does not seem to be a computationally efficient way to do this.

How well could the candidate elimination algorithm learn text analysis rules using CRYSTAL's representation? A tiny example from the Management Succession domain will help illustrate this. Figure 8.5 has four training instances. For clarity, only the motivating sentence is shown rather than the CRYSTAL instance. The target concept here is *Person_In* in the subject and *Position* in the direct object.

1. (pos) Mr. A was named chairman.
2. (pos) Mr. C will be the new president of XYZ Inc.
3. (neg) The board of ABC Corp. ousted the chairman.
4. (neg) Mr. C has been chairman since 1983.

Figure 8.5 Four instances to illustrate the candidate elimination algorithm

Figure 8.6 shows the concept definitions in the S set and the G set after instance 1 and instance 2 have been read. The S set has a single concept definition that is identical to CRYSTAL's unification of the initial definitions from instances 1 and 2. I am showing this in a simplified format, but s_1 is actually a concept definition. The G set has a concept definition with no constraints, since there are no negative instances to avoid yet.

S set:
s_1 Subj has term "Mr." and class <Person Name>,
Obj has the class <Corporate Post>
G set:
g_1 A concept definition with no constraints

Figure 8.6 The S set and G set after the first two instances have been read.

Figure 8.7 shows S and G after instance 3 has been read. The S set is not changed by a negative instance, but the G set must now be specialized to avoid a negative instance with a <Corporate Post> in the direct object.

S set:
s_1 Subj has term "Mr." and class <Person Name>,
Obj has the class <Corporate Post>
G set:
g_1 Subj has term "Mr." and class <Person Name>

Figure 8.7 The S set and G set after three instances have been read.

After reading the fourth instance, the version space collapses. The new negative instance has "Mr." and class <Person Name> in the subject, which means that g_1 no longer covers only positive instances. There are no concept

definitions that cover both positive instances while avoiding both negative instances.

These four training instances are not inconsistent. Mitchell refers to this situation as a *disjoint* concept that cannot be expressed by a single concept definition. The candidate elimination algorithm copes with disjoint concepts by maintaining multiple version spaces and also allows for some amount of disjunction in the representation language.

CRYSTAL would arrive at a rule base with two concept definitions, one to cover each of these positive instances. With more training, these concept descriptions would become generalized, but separate concept definitions would still be needed to cover instance 1 and instance 2.

CRYSTAL does something akin to building an S set when it generalizes a seed instance. CRYSTAL's similarity metric finds the most useful positive instance to consider next in creating this generalization. By finding a similar positive instance, CRYSTAL continues working on the same portion of a "disjoint" concept.

The main function of the G set is to form a compact represention of the negative training instances. As the example from Figure 8.5 suggests, a G set is quite fragile given CRYSTAL's representation and would collapse easily. CRYSTAL opts to consult the negative training instances directly when evaluating a proposed generalization, rather than build a G set.

8.4 Instance Based Learning

Both CRYSTAL and instance based learning (IBL) use a distance metric in a critical step in the algorithm, although the distance metric plays a

fundamentally different role. IBL uses this metric directly in classifying new instances by finding the most similar training instances or *exemplars*. Some systems look for a single most similar exemplar, while others let the "k nearest neighbors" vote on a classification.

The distance metric in CRYSTAL is used as a search heuristic, and not used to classify instances. The class (the concept being learned) is already known and CRYSTAL applies the similarity metric only to instances of that class.

Two IBL algorithms with different strategies for handling noisy training instances are David Aha's IB3 [Aha *et al.* 1991] and Scott Cost and Steven Salzberg's PEBLS [Cost and Salzberg 1993]. Each of these algorithms tabulates classification performance statistics on each exemplar to reduce the effect of noisy training instances.

IB3 bases its classification of new instances only on instances that pass a statistical significance test as reliable classifiers. Those with performance significantly worse than chance are discarded. IB3 uses the confidence thresholds of its significance tests to control the rate of acceptable classification errors in somewhat the same way that CRYSTAL uses its error tolerance parameter. IB3 also saves space by only keeping misclassified instances, which are the most likely to be useful in refining a concept boundary.

PEBLS keeps the entire training set as exemplars and assigns to each a weight based on its performance in classifying the rest of the training instances. PEBLS classifies a new instance by computing its distance to each of the exemplars and then multiplying this similarity measure by the weight assigned

to the exemplar. This increases the effective distance to unreliable exemplars and limits their influence in classification.

Both CRYSTAL and instance based learning use a bottom up approach to define local regions in instance space. CRYSTAL explicitly describes a region boundary by the constraints in a concept definition. The region expands from a seed instance until further growth is inhibited by negative instances. Instance based learning implicitly defines a region surrounding a stored training instance. The region contracts when a nearby instance is added to the exemplars and expands when a nearby exemplar is discarded. In PEBLS a large weight causes the region affected by an exemplar to shrink.

Training IB3 or PEBLS is less computationally expensive than it was for the covering algorithms. Compiling performance statistics on each exemplar requires computing the distance between it and each of the n training instances. The published versions of IB3 and PEBLS apparently consider all possible attributes a when computing the distance. This gives $O(an)$ computations for each of $O(n)$ potential exemplars, for an overall time complexity of $O(an^2)$.

If an instance representation and distance function such as CRYSTAL's were used, the only attributes considered would be those actually found in the two instances being compared. This would eliminate the dependency on a and give IB3 and PEBLS a time complexity of $O(n^2)$.

The memory required for instance based learning during training is proportional to n, as it is for CRYSTAL. After training, however, CRYSTAL can discard the training instances and use the set of learned rules, which will be much smaller in general than the full set of instances. IB3 discards much of

the training set, but PEBLS retains all training instances, which can become expensive in terms of memory.

8.5 Decision Tree Algorithms

A well known family of machine learning algorithms is top down induction of decision trees. I will use Ross Quinlan's C4.5 [Quinlan 1993] as a representative of decision tree algorithms and use Giulia Pagallo and David Haussler's GREEDY3 algorithm [Pagallo and Haussler 1990] to represent decision lists, which are a special case of decision trees.

8.5.1 C4.5

A decision tree algorithm such as C4.5 begins with an empty tree and recursively adds tests at each tree node to partition the instance space. The key step in top down induction of a decision tree is to select a feature at each node whose values best separate different classes into different partitions. Some decision tree algorithms such as OC1 [Murthy *et al.* 1994] also allow the test at a node to be a linear combination of features. This increases the computational cost of selecting a test as each node.

Many decision tree algorithms, including C4.5, base the feature selection metric on the information-theoretic measure, Shannon entropy, shown in Section 8.3.2. Another basis for a "goodness of split" metric is the *Gini* diversity index [Breiman *et al.* 1984].

$$Gini\_index = \sum_{j \neq i} p_i p_j$$

Experiments by John Mingers suggest that the particular feature selection metric used may not be critical to performance [Mingers 1989], although the right choice of metrics can increase performance for a particular data set.

With its recursive partitioning, top down decision tree induction tends to fragment the instance space. Nodes near the leaves of the tree are often based on a small number of training examples and have low predictive accuracy. Decision tree algorithms have a mechanism that serves much the same function as CRYSTAL's error tolerance parameter. Pruning away branches of a decision tree will often improve classification accuracy, although no distinction is generally made between optimizing for recall and optimizing for precision.

How well can decision trees handle extremely large number of attributes? The C4.5 algorithm tabulates statistics on how often each of a attributes is found in instances of each class (e.g. positive or negative). Each level of the tree has up to n training instances, which results in computation time of $O(an)$ for each level of the tree. The total computation time is $O(adn)$, where d is the tree depth.

The $O(adn)$ computations required by C4.5 are much simpler operations than CRYSTAL's $O(rn)$ operations. C4.5's basic operation is to tabulate how often a value in a feature vector is associated with particular classes. CRYSTAL's basic operation is to test a proposed concept definition against a training instance, which can be much more expensive than C4.5's basic operation.

In an ideal, noise-free instance space the tree depth d would depend only on the underlying concept being learned. Tree depth is bounded by a, since no path may have more than a tests. If a is greater than n (which may actually be

the case in learning text analysis rules), a badly skewed tree may have depth of $O(n)$. This would make the time complexity $O(an^2)$. This is a pessimistic analysis, since in practice d tends to grow more slowly than n. Growth of tree depth as n increases is analogous to the growth of the number of rules in CRYSTAL and other covering algorithms.

C4.5 represents training instances as a feature vector with length $O(a)$. All n instances must be repeatedly consulted while building a decision tree, which gives a space requirement of $O(an)$.

8.5.2 GREEDY3

A variant of decision trees is the decision list, introduced by Ronald Rivest [Rivest 1987]. Each node of a decision list has a test that splits the instance space into exactly two partitions, at least one of which must be a leaf node. Thus each node separates off a set of training instances from the remaining instances. This has been characterized as "separate and conquer" as opposed to a decision tree's "divide and conquer" strategy.

Decision lists are able to reach a leaf node with every test by using a multivariate test, one that evaluates multiple features. This makes the decision list equivalent to a list of rules, each rule having several constraints.

The GREEDY3 algorithm [Pagallo and Haussler 1990] uses a top down approach to building the test at each node of a decision list. GREEDY3 adds one feature at a time to the test, selecting the feature that has the highest probability of being found in a positive instance. This requires tabulating how often each literal is associated with positive instances.

When enough features have been added to the test that it covers only positive instances, GREEDY3 adds the test as a node of the decision list. Instances covered by the test are removed from training set in much the same way that a covering algorithm eliminates instances covered by each new rule. GREEDY3 continues creating additional nodes until the decision list cover all positive instances. At that point GREEDY3 terminates the decision list with a leaf node for "negative".

Like decision tree algorithms, GREEDY3 includes a pruning step to improve performance. Pruning can remove the last feature that was added to a rule or remove the rule entirely if it does not improve performance on an independent set of instances.

GREEDY3 has much in common with C4.5 and other top down decision tree algorithms. It operates in a top down fashion, considers all features exhaustively when selecting a feature to add to a node, and uses a selection metric based on how each feature performs on the training instances.

The computation required to select a feature to add to a rule is $O(an)$, where a is the number of features (attributes) and n is the number of training instances. Let r be the number of rules in the decision list and k be the number of features in each rule. Time complexity to build the entire decision list is $O(arn)$, if k is treated as a constant.

In an extremely pessimistic worst case, $r = O(n)$ and $k = O(a)$, giving time complexity of $O(a^2n^2)$. In practice, the r in GREEDY3's time complexity is likely to have a growth rate similar to the r in CRYSTAL's time complexity of $O(rn)$.

Could GREEDY3 be re-implemented to avoid exhaustive consideration of features? Like C4.5 and CN2, there seems to be no way to avoid this. GREEDY3 does not have a designated "seed" instance to limit the selection of features.

8.5.3 An Experiment with C4.5

I conducted some preliminary experiments in applying C4.5 to learn text analysis rules. These experiments were abandoned due to the excessive computation time taken when even 30% of the Hospital Discharge texts are used as training. I built C4.5 trees for the concept *Symptom,Present*.

Each C4.5 instance contained features representing the same syntactic-lexical or syntactic-semantic constraints as a CRYSTAL instance. Boolean features were created as described in Section 8.2. Each feature represents a particular constraint (e.g. Head terms) on a particular word or semantic class in a particular syntactic constituent (e.g. subject).

Each CRYSTAL instance was turned into multiple C4.5 instances, each designating one of the syntactic constituent in the instance. If, for example, a CRYSTAL instance had a subject, verb, direct object, and two prepositional phrases, it was turned into five C4.5 instances. The first instance had a positive classification if the subject contained *Symptom,Present*. The second instance was positive if the verb contained *Symptom,Present*, and so forth.

Section 8.1 discusses why a classifier needs multiple copies of a CRYSTAL instance. To avoid this five-fold increase in training instances, a separate C4.5 tree was built for each constituent: SUBJ, VERB, OBJ1, OBJ2, and PP[5].

[5]Subject, verb, direct object, indirect object, and prepositional phrase, respectively.

The set of possible features was pruned by discarding any feature not found in at least ten training texts. Only these high frequency features were used in the C4.5 feature vectors. Even after eliminating the low frequency features, each tree still had extremely large feature vectors. OBJ1 had 2,356 instances with 2,268 features; PP had 3,389 instances with 4,308 instances; and so forth.

The main difficulty in running C4.5 with such large feature vectors turned out to be the space requirement. The instances for OBJ1 took 27Mb of memory on an DEC ALPHA 3000 workstation. This fit into resident memory and the OBJ1 decision tree was induced in 18 minutes. The instances for the PP tree took 67Mb with only half of that as resident memory. The computer ran so inefficiently from continual swapping that C4.5 was still selecting a test for the second tree node after 41 hours.

The trees were finally completed on a different machine that handled the memory swapping more efficiently. Recall and precision was comparable to CRYSTAL's on this data set. C4.5 had recall 34 at precision 66, while CRYSTAL had recall 36 at precision 61[6].

Not too much can be made of comparing this single data point. Average recall and precision were two points higher for C4.5 for this concept. C4.5 was also run for two concepts trained on the Management Succession data. Average recall and precision for C4.5 was 6 points lower than CRYSTAL for one concept and 10 points higher for the other[7].

[6]This was run on with older version of syntactic analysis that did not label phrases as affirmative or negative. CRYSTAL was run with exception learning at error tolerance 0.20 and min-coverage 5.

[7]These were concepts from an earlier set of experiments, *Person Name* and *Organization Name*.

The trees learned by C4.5 show a strong bias toward single-feature rules. The trees were so badly skewed that they were essentially decision lists. The root node of the OBJ1 tree was a test for "gravida" as a modifier term in the direct object. This identified 26 out of the 2,356 training instances as positive. The next node tested for the term "mass" in the direct object, which identified another 15 instances as positive.

The PP tree showed the same behavior. The root node identified 12 positive instances out of the 3,389 training instances. The test was for the term "shortness of breath" in the prepositional phrase.

A re-implementation of C4.5 could reduce its memory footprint by using a sparse vector representation that explicitly records only the true-valued features. This would allow C4.5 to handle larger training sets. In fairness to C4.5, CRYSTAL would face the same problem of exceeding memory capacity if the Hospital Discharge corpus grew much larger.

8.6 Contrast of CRYSTAL and Other ML Algorithms

Some of the differences between CRYSTAL and the other algorithms described in this chapter are superficial. CRYSTAL's instance representation is a natural way to encode the syntactic, semantic, and lexical features of a clause. However, much the same information can be expressed in terms of Boolean features or predicate calculus, depending on the input requirements of the machine learning algorithm. Paths in a decision tree are functionally equivalent to rules in a decision list or to concept descriptions in CRYSTAL and other covering algorithms.

Other differences are more fundamental to how the algorithm operates. Many of the algorithms described in this chapter operate in a top down fashion. They start from very general rules with a single constraint and then specialize the rule by adding further constraints. This applies to A^q, CN2, C4.5, and GREEDY3. CRYSTAL works from the opposite direction, beginning with maximally specific descriptions and generalizing by relaxing constraints.

This gives CRYSTAL a different "learning bias" from the top down algorithms. CRYSTAL tends to relax constraints just enough to cover a group of positive training instances, even if some of the constraints could be dropped without covering any additional negative instances. Top down algorithms add just enough constraints to avoid negative instances, even if further constraints could be added without excluding any positive instances.

Instance based learning operates from the bottom up, but has a radically different way to classify instances. IBL does not create anything analogous to rules. For this reason, it is hard to characterize the difference in learning bias between CRYSTAL and IBL. IBL keeps both positive and negative instances as exemplars that define positive or negative regions of instance space by proximity. CRYSTAL define a region of instance space by constraints in a concept definition. Portion of instance space not covered by a concept definition are negative by default.

Candidate elimination algorithm builds one boundary of the version space from the top down and another from the bottom up at the same time. Vere's algorithm operates from the bottom up. Both of these algorithms identify *all* concept descriptions that are consistent with the training data, which makes learning bias less of an issue.

It is not clear that one learning bias is more appropriate than another in general. A bias that gives high performance on one data set will not necessarily be the best on another data set. The difference in bias will be most noticeable when the training data is noisy or insufficient.

The greatest contrast between CRYSTAL and the other machine learning algorithms described here is in time complexity. The covering algorithms A^q and CN2 require $O(asrn)$ computations, where a is the number of attributes and s is the "star" size used in a beam search. A^q could be re-implemented to consider only those features found in the seed instance, which would reduce its time complexity to $O(srn)$. GREEDY3 requires $O(arn)$ computations. The r in these time complexity analyses, the number of "rules", has a slightly different meaning for each algorithm.

Inducing a C4.5 decision tree requires $O(adn)$ computations, where d is the decision tree depth, somewhat analogous to r in the above time complexities. Although C4.5 cannot avoid computation time proportional to a, its basic operation is quite cheap: incrementing a counter based on a feature vector value. This allows it to handle fairly large feature sizes before it becomes overwhelmed.

Training requires $O(an^2)$ computations for the instance based learning algorithms IB3 and PEBLS and could be re-implemented to take $O(n^2)$. Computation time for the candidate elimination algorithm grows with the square of the boundary set sizes, which may become quite large.

CRYSTAL requires $O(rn)$ computations where r is the number of rules learned and n is the total number of training instances. This is bounded by $O(pn)$, where p is the number of positive training instances. CRYSTAL's time

and space requirements are independent of a, allowing it to handle extremely large feature sets easily.

CHAPTER 9

CONCLUSIONS

CRYSTAL's richly expressive representation language and its efficient algorithm for navigating extremely large feature spaces are two sides of the same coin. Formulating text analysis rules for unrestricted natural language requires flexibly combining syntactic, semantic, and lexical evidence. An expressive representation can pose problems for a learning algorithm, however, and requires an efficient search strategy that is not misled by irrelevant features.

The few existing systems that learn text analysis rules from training examples, all have limited rule representation. Rules may have semantic constraints, but not lexical constraints, on phrases to be extracted. Certain sentence elements are always included in the rules and others never included.

CRYSTAL's approach is to include any lexical or semantic constraints on any syntactic constituent of the instance. A corpus of several hundred texts may contain enough distinct words that this leads to thousands of features[1].

CRYSTAL's learning algorithm can handle such a large number of features because of space and time requirements that do not depend on the size of the feature set. CRYSTAL has a bottom up strategy that begins with a

[1]When instances from a training set of 150 Hospital Discharge texts were converted into Boolean features, there were over 4,000 such features. This was after discarding any feature not found in at least ten texts.

seed instance and guides generalization by finding the most similar positive instance. Features not found in the seed instance or in the similar instance are not considered.

Relaxing constraints to unify with a similar positive instance has the result of quickly dropping features that are simply accidental to the seed instance, while tending to retain essential features. CRYSTAL is not slowed down by irrelevant features. Including as many features as possible that *might* be useful increases CRYSTAL's performance and robustness with respect to noisy data.

9.1 Contributions

The major contributions of this work are presented in the next three subsections:

1. Learning high quality text analysis rules
2. An efficient covering algorithm
3. The impact of expressive representation

9.1.1 Learning High Quality Text Analysis Rules

CRYSTAL demonstrates that high quality text analysis rules
can be learned from examples, rules which:
- a. approach the performance of hand-coded rules
- b. are robust in the face of noise and inadequate features
- c. require modest training size

CRYSTAL has been tested on multiple domains. Its performance, in terms of recall and precision, varies according to a number of factors: the degree of

regularity in how references to the target concept are worded, the amount of training, and the amount of noise in the data.

Experiments were conducted that compare CRYSTAL's performance with that of hand-coded rules. CRYSTAL achieved over 90% the average recall and precision of hand-coded rules on input that had high quality semantic tagging. On a version of one domain with coarser semantic tagging, CRYSTAL had over 80% the performance of hand-coded rules.

One aspect of CRYSTAL that allows it to compensate for noisy training data is CRYSTAL's expressive representation. When one source of evidence, such as semantic class assignment is unreliable, CRYSTAL falls back on other evidence. Experiments were done that compare input with fine-tuned semantic tagging to input with semantic tagging based on a generic thesaurus. CRYSTAL relied more on exact word constraints with the lower quality semantic tagging and generated twice as many rules.

CRYSTAL has an error tolerance parameter that allows it to accommodate noise in the data. Together with a minimum coverage parameter, this gives the user a knob to manipulate a trade-off between recall and precision.

The amount of training data required is a concern for a supervised learning algorithm such as CRYSTAL. A domain expert must select a corpus of representative texts and then label each reference to the concepts of interest for the domain. CRYSTAL's job is to learn rules that will imitate these human annotations on previously unseen texts.

The Management Succession corpus used in this thesis took about one week of human effort to annotate and the Hospital Discharge corpus took about three weeks. While this is not an insignificant investment of time, I feel that

I can call this a modest amount of training. The training corpus for Hospital Discharge consisted of less than 150,000 words. Statistical corpus-based techniques often require tens of millions of words of training [Church *et al.* 1991].

It is a combination of the limited domain and the use of semantic class information that allow CRYSTAL to work with such a comparatively small amount of training. David Fisher and Ellen Riloff have shown that statistically significant co-occurrence frequencies can be derived from a small corpus in a limited domain [Fisher and Riloff 1992].

The amount of training can be viewed as modest from another point of view. Developing a set of rules by hand also requires a set of annotated examples to guide development for all but the simplest of information extraction tasks. This means that CRYSTAL's training corpus is not an additional expense over a manual engineering approach.

9.1.2 An Efficient Covering Algorithm

> CRYSTAL presents a covering algorithm control strategy that navigates efficiently in extremely large feature spaces:
>
> a. time and space complexity independent of the feature size
> b. a greedy approach that can be improved very little by using more extensive search

CRYSTAL belongs to the family of machine learning algorithms known as covering algorithms. This family of algorithms generates a set of rules from training examples such that each rule covers as many positive instances as possible, while avoiding negative instances. New rules are added to the rule base until all positive training instances have been covered.

What distinguishes CRYSTAL from other covering algorithms, and from top down decision tree and decision list induction, is its ability to handle extremely large numbers of features. Rather than considering all possible attributes exhaustively, CRYSTAL considers only the attributes found in a seed instance and in one similar instance at a time. This give CRYSTAL time and space requirements that do not depend on the total number of features.

CRYSTAL's efficient algorithm permits it to have an expressive rule representation with no limit on the number of features. CRYSTAL's features include any of thousands of exact words in any of several syntactic roles[2]. Large feature sets arise naturally when applying machine learning techniques to the analysis of unrestricted text.

CRYSTAL is also efficient because of its "greedy" control strategy. At each step in generalizing a concept definition, CRYSTAL finds the relaxation that appears best and never goes back to considers alternate possibilities.

Experiments were conducted that use a beam search version of CRYSTAL. Rather than commit to a single possible generalized definition, a beam search maintains the best w definitions found so far, where w is the beam width. A large beam width increases the amount of search effort and increases the likelihood that CRYSTAL will find optimal generalizations from each seed instance.

A large beam width produces compact rule sets, but generally produces no improvement in average recall and precision on a blind test set. Increasing the beam width raises recall but lowers precision. The more extensive search

[2]e.g. SUBJ-Head_Term-ANGINA and PP-IN-Modifier_Term-SUBSTERNAL in the Hospital Discharge domain.

results in concept definitions that seem reliable on the training set, but are often overgeneralized on the test set.

Other covering algorithms, A^q and CN2, rely on a beam search approach. CRYSTAL is unique among covering algorithms in attaining high performance from beam width of one.

9.1.3 The Impact of Expressive Representation

> CRYSTAL demonstrates that expressive representation is essential for high performance, robust text analysis rules.

A richly expressive representation is not guaranteed to improve performance. A larger feature set results in a higher dimensionality of the instance space. This could make learning more difficult and increase the likelihood that a learning algorithm will be misled.

Empirical studies show that CRYSTAL's expressiveness has a positive impact on performance. Experiments compared the basic CRYSTAL system with other versions that included various restrictions to CRYSTAL's representation as well as one enhancement. The enhancement was to add exceptions to CRYSTAL's rules. The restrictions were each motivated by aspects of CRYSTAL's representation that are missing in other published systems that learn text analysis rules.

Taken singly, the enhancement or restrictions had a small effect on performance. For some concepts in some domains, learning exceptions increased the average of recall and precision slightly. Restrictions tended to hurt perfor-

mance, although in some cases a restricted representation had slightly better performance on some concepts than the baseline system.

When four restrictions were applied at once to CRYSTAL's representation, the results were generally disastrous. Recall plummeted in most cases, although it was essentially unchanged for a few concepts. While simple rule representation may be sufficient for some concepts in some domains, CRYSTAL's expressiveness is needed for high performance in general and increases CRYSTAL's ability to handle noise and limited training data.

9.2 Future Work

I will outline two areas of future exploration. The first includes modifications to CRYSTAL itself. The second suggests ways to improve CRYSTAL's utility as an information extraction module.

9.2.1 Enhancements to CRYSTAL

CRYSTAL is a fairly mature system, as software goes, having been tested on several domains over the course of two years. The most obvious attempts to enhance the basic CRYSTAL algorithm have already been tried: learning exceptions to rules and increasing the search effort with a beam search. Several minor changes to the system remain to be explored that each promise small improvements in performance.

CRYSTAL currently selects seed instances in an arbitrary order and often makes a few false starts on the way to learning a high coverage concept definition. There would be fewer irrelevant features to lead CRYSTAL astray if the

seed instances were sorted so that those with fewest features are selected first. This may result in a smaller set of rules with somewhat higher performance.

The current method of learning exceptions is also not satisfactory. When adding exceptions to a proposed definition brings it back within error tolerance, CRYSTAL tries to relax constraints further. This tends to result in concept definitions that are badly overgeneralized before exceptions are added. The net effect is to raise recall and lower precision.

I plan to experiment with alternative methods of adding exceptions to rules. CRYSTAL could add exceptions in a separate step after it has generalized a concept definition. Generalization would halt as in the basic algorithm and return a definition that is within error tolerance without exceptions. At that point CRYSTAL would learn a set of exceptions sufficient to exclude the training errors covered by the definition. Even if only some of the exceptions apply to blind test instances, this would raise precision somewhat without lowering recall.

Hardly any experimentation has been done on the distance function used to find the most similar instance. My assumption has been that getting exactly the right distance function is not critical, but this has not been tested. Several parameters have been built into the distance function and others could be devised to bias which instance is selected as most similar. Should CRYSTAL have a tendency to consider semantic similarity as more important than words in common? Are similarities in extracted phrases more important than similarities in other phrases? Is the verb more important than other sentence elements?

While doing experiments in which I built rules by hand, I noticed that I had a strong bias towards rules with only a few constraints. CRYSTAL's bottom up approach has a strong tendency to produce rules with many more constraints. It would be interesting to change CRYSTAL's bias by adopting a top down version of CRYSTAL.

CRYSTAL would begin with a seed instance and consider all possible single-constraint rules derived from features of the seed instance. If all of these covered too many negative instances, CRYSTAL would add additional constraints, one at a time. This would probably require a beam search like that of the A^q covering algorithm for best results.

The advantage of a top down CRYSTAL is that its learning bias might give better performance in terms of recall and precision on some data sets. The disadvantage is more certain than the possible advantage. Performance in terms of computation time would be one or two orders of magnitude greater than the current CRYSTAL. For some information extraction tasks it is worthwhile to spend hours rather than minutes of CPU time if this produces better rules.

A last area for improvement is in efficiency of implementation. In particular, space efficiency of representing an instance becomes vital if CRYSTAL is to hold large training sets resident in memory.

9.2.2 Versatility as an Information Extraction Module

CRYSTAL does not function in a vacuum. Its performance in an information extraction (IE) system depends on the syntactic and semantic analysis of its input. Its contribution to the IE system also depends on the granularity of its input and output.

It is vital to design a level of syntactic analysis that gives CRYSTAL the most useful input possible. The flat structure of CRYSTAL's instance representation, which does not allow nested structures, make this a challenge. The input text in Figure 9.1 illustrates this problem.

> Nicholas Mihalas, SmartCard's 51-year-old president and chief executive officer, was named to the additional post of chairman, succeeding Mr. Lessin, whose resignation was effective last Monday.

Figure 9.1 A text with appositives and relative clauses

This sentence has a syntactic complexity that poses a severe challenge to the flat structure of CRYSTAL's instance representation. Nicholas Mihalas is separated from "was named" by a lengthy appositive. The relationship between Mihalas and Mr. Lessin is indicated by a reduced relative clause "Nicholas Mihalas ... succeeding Mr. Lessin". Additional evidence that Lessin is a *Person_Out* comes from a relative clause "whose resignation ...".

CRYSTAL needs the entire sentence presented as a single instance if it is to find the relationship between Mihalas and Lessin. Figure 9.2 shows the style of syntactic analysis used for experiments in this thesis.

SUBJ:	Nicholas Mihalas, SmartCard 's 51-year-old president and chief executive officer
VERB:	was named
PP:	to the additional post of chairman
REL-VERB:	succeeding Mr. Lessin, whose resignation was effective last Monday.

Figure 9.2 Syntactic analysis that lumps an appositive with the subject and lumps together words in relative clauses

This analysis lumps together a person name, a company name, and two positions in the subject. Even if CRYSTAL learns a concept definition that correctly extracts a *Person_In* from the subject, it does not specify which part of the subject contains the relevant information. This is left to later processing in an IE system.

The REL-VERB in this instance (relative clause attached to the verb) is also presented as a bag of undifferentiated words. Relative clauses may be long and contain embedded clauses, as this one does.

Given instances at the proper level of granularity, CRYSTAL could learn rules that identify exactly which simple noun phrase to extract. The most promising approach is to present CRYSTAL with multiple views of the input text. The high level view of the sentence is given in Figure 9.2. An additional instance would break apart the relative clause as shown in Figure 9.3.

SUBJ:	Nicholas Mihalas, SmartCard 's 51-year-old president and chief executive officer
VERB:	succeeding
OBJ:	Mr. Lessin
REL-OBJ:	whose resignation was effective last Monday.

Figure 9.3 A second view of the sentence that breaks apart the REL-VERB

This second view of the input shows the syntactic relationship between "succeeding" as a verb and "Mr. Lessin" as a direct object. Heuristics are needed to allow a relative clause to inherit its subject from the main clause. This second instance not only provides better syntactic information, but allows a concept definition to pinpoint "Mr. Lessin" as the *Person_Out*. A

third instance would also be created to break apart the relative clause "whose resignation ..." and its inherited subject "Mr. Lessin".

A different approach is needed for appositives and lists such as that found in the subject "Nicholas Mihalas, SmartCard 's 51-year-old president and chief executive officer". A second pass of CRYSTAL could learn text analysis rules at the level of noun phrase analysis. For this, the complex noun phrase would be presented as a list of simple noun phrases separated by delimiters.

NP:	Nicholas Mihalas
DELIM:	%COMMA%
NP:	SmartCard
DELIM:	's
NP:	president
DELIM:	and
NP:	chief executive officer

Figure 9.4 An instance for noun phrase analysis to identify relevant information within a complex noun phrase

To make best use of CRYSTAL for noun phrase analysis, the concept definitions would need to include an ordering constraint. Order of constituents is implicit in the names SUBJ, VERB, and OBJ, but CRYSTAL would need explicit constraints to distinguish the NP that immediately precedes the delimiter "'s" from the NP that follows the delimiter.

Multiple constituents with the same name also raise problems for the low-level CRYSTAL functions that compute distance. There will be many possible mappings between two instances with several NP's. Some of these mappings will preserve ordering of the NP's and others will not. The distance between the two instances is ideally based on the mapping that optimizes similarity.

CRYSTAL needs a scheme to find a nearly optimal mapping that takes into account ordering constraints as well as pair-wise similarity of constituents.

9.3 Implications for System Development

At the heart of this research lies the problem of knowledge acquisition for domain-specific text analysis. Each new domain or information need requires a new set of text analysis rules to be learned that identify references to concepts important to that domain.

These rules depend critically on the quality of the supporting knowledge sources, such as the semantic tagging of individual words. At one extreme, there may be a semantic tag that corresponds perfectly with a target concept. In this case the text analysis rules can be expressed simply in terms of the corresponding semantic tag and are trivially easy to learn. Suppose our target concept is *Person,Name* and every person name in the input has the semantic tag <Person Name>.

At the other extreme, there may be no semantic tags at all for a new domain, or an extremely weak correlation between semantic tags and the target concepts. In this case, much more complicated text analysis rules are needed. A large set of rules will be needed to account for the many contexts in which a concept occurs, often expressed as exact word constraints. Generating a set of text analysis rules is difficult whether done manually or using machine learning techniques, when the semantic tagging has not been tailored to the target concepts.

The starting point in developing an information extraction system for a new domain is typically between these two extremes. A generic semantic

lexicon and semantic hierarchy may be available, but one that only roughly fits the target concepts. The semantic hierarchy may fail to make necessary discriminations and the semantic lexicon may lack coverage of important terms for the target concepts. Tailoring a semantic lexicon and semantic hierarchy to a particular information need is a time consuming manual task.

In many cases the target concepts are not clearly known in advance. Early development will be based on a set of concepts that are assumed to be useful and easy to extract automatically. The exact boundaries of these target concepts may be refined later. The first pass of text annotation is bound to contain inconsistencies as the annotators come to grip with what exactly should count as a positive example of each concept.

All of this creates a noisy situation when adapting a system to a new information extraction task. Text analysis rules must operate robustly in the face of limited coverage by a semantic lexicon, poor fit of a semantic hierarchy, imperfect syntactic analysis, and inconsistent training annotations.

CRYSTAL demonstrates that text analysis rules can be learned automatically, even when faced with all these difficulties. A robust, fully automatic tool such as CRYSTAL allows good system performance from the beginning. Later, when semantic tagging and syntactic analysis have been customized to the information need and inconsistent annotations have been minimized, CRYSTAL will take advantage of these refinements to boost system performance.

APPENDIX A

TABLES OF MANAGEMENT SUCCESSION RESULTS

Concept	Pos.	Min.Cover 2			Min.Cover 10			Min.Cover 25		
	Train	R	P	Avg	R	P	Avg	R	P	Avg
In	123	58.3	67.9	63.1	47.6	74.4	61.0	28.2	69.2	48.7
	243	62.6	69.8	66.2	55.1	75.4	65.3	40.1	77.1	58.6
	505	71.5	66.9	69.2	66.1	71.6	68.9	57.7	73.2	65.4
Out	129	48.9	62.3	55.6	30.2	72.7	51.5	0.0	0.0	0.0
	264	55.1	68.2	61.6	47.3	77.8	62.5	16.6	69.3	42.9
	529	63.4	68.9	66.1	57.3	77.8	67.5	48.9	82.0	65.4
Post	191	59.1	65.7	62.4	46.6	71.4	59.0	28.1	65.8	46.9
	383	64.7	66.9	65.8	57.9	71.8	64.8	35.7	71.8	53.8
	772	72.1	66.4	69.2	67.4	69.9	68.7	60.4	74.4	67.4
Org	154	39.5	58.0	48.8	21.1	68.1	44.6	8.9	43.6	26.2
	305	44.2	59.3	51.7	31.1	66.2	48.6	14.8	73.4	44.1
	617	53.8	63.3	58.5	43.9	71.5	57.7	22.6	77.0	49.8
In,Out	33	56.9	78.5	67.7	35.0	62.6	48.8	0.0	0.0	0.0
	69	64.8	79.6	72.2	55.2	81.8	68.5	27.0	54.8	40.9
	145	75.3	76.6	75.9	68.8	76.8	72.8	56.7	77.0	66.9
In,Post	94	53.6	77.8	65.7	38.3	85.2	61.7	28.6	78.5	53.5
	190	60.6	74.1	67.3	49.2	78.5	63.9	38.2	79.8	59.0
	383	69.6	71.5	70.6	61.9	73.5	67.7	48.1	76.5	62.3
In,Org	72	33.5	64.4	48.9	22.2	76.1	49.1	1.9	7.5	4.7
	142	38.5	65.9	52.2	24.7	75.4	50.1	17.5	72.5	45.0
	290	49.7	68.5	59.1	33.5	76.2	54.9	19.1	77.9	48.5
Out,Post	104	43.1	66.6	54.8	23.2	73.2	48.2	0.0	0.0	0.0
	213	50.5	69.2	59.8	38.9	80.1	59.5	8.7	60.2	34.5
	424	61.7	72.0	66.8	52.8	81.6	67.2	38.8	86.0	62.4

Table A.1 Management Succession results at error tolerance 0.20 (continued on next page)

Concept	Pos.	Min.Cover 2			Min.Cover 10			Min.Cover 25		
	Train	R	P	Avg	R	P	Avg	R	P	Avg
Out,Org	81	27.0	59.4	43.2	6.3	40.0	23.1	0.0	0.0	0.0
	165	35.1	63.0	49.1	19.6	70.4	45.0	0.5	7.9	4.2
	327	46.5	71.0	58.8	36.5	81.0	58.8	15.8	90.4	53.1
Post,Org	136	38.7	65.9	52.3	20.0	72.9	46.5	9.9	38.4	24.2
	271	45.2	66.3	55.7	29.9	69.3	49.6	16.0	71.9	44.0
	548	55.3	70.9	63.1	43.3	75.8	59.5	20.9	76.3	48.6
In,Out,Post	17	25.7	88.7	57.2	0.0	0.0	0.0	0.0	0.0	0.0
	37	39.1	87.3	63.2	18.1	67.3	42.7	0.0	0.0	0.0
	76	53.2	82.5	67.9	49.0	83.7	66.3	11.7	32.8	22.3
In,Out,Org	12	7.8	67.8	37.8	0.0	0.0	0.0	0.0	0.0	0.0
	25	13.2	60.7	37.0	0.0	0.0	0.0	0.0	0.0	0.0
	52	28.4	81.1	54.8	19.7	86.7	53.2	0.0	0.0	0.0
In,Post,Org	65	30.0	69.8	49.9	22.2	79.5	50.9	2.1	7.5	4.8
	130	36.8	69.3	53.1	24.7	74.8	49.8	19.4	69.5	44.5
	266	45.5	71.8	58.7	30.5	73.8	52.1	21.5	77.4	49.4
Out,Post,Org	73	22.4	65.9	44.1	5.7	31.6	18.6	0.0	0.0	0.0
	151	32.8	66.5	49.7	16.6	72.4	44.5	0.5	8.3	4.4
	300	47.0	74.1	60.5	37.6	85.0	61.3	10.4	85.3	47.8
In,Out,Post,Org	9	5.5	38.6	22.0	0.0	0.0	0.0	0.0	0.0	0.0
	22	11.6	72.8	42.2	0.0	0.0	0.0	0.0	0.0	0.0
	46	26.0	86.7	56.3	25.0	86.7	55.8	0.0	0.0	0.0

Table A.2 Management Succession results at error tolerance 0.20 (continued from previous page)

APPENDIX B

TABLES OF HOSPITAL DISCHARGE RESULTS

Concept	Pos.	Min.Cover 2			Min.Cover 10			Min.Cover 20		
	Train	R	P	Avg	R	P	Avg	R	P	Avg
Present	393	33.0	57.4	45.2	22.2	74.2	48.2	18.0	82.0	50.0
	1192	39.8	61.0	50.4	29.9	74.0	51.9	24.0	79.8	51.9
	2741	44.0	63.9	54.0	35.8	74.3	55.0	29.6	78.8	54.2
Absent	444	61.6	73.1	67.4	51.5	79.8	65.6	35.5	84.6	60.0
	1312	70.0	76.0	73.0	63.0	79.4	71.2	56.3	82.0	69.1
	3025	73.6	77.1	75.3	68.3	79.9	74.1	63.8	81.3	72.5
Confirmed	208	53.4	66.5	59.9	42.4	72.3	57.4	33.2	75.4	54.3
	625	57.4	67.5	62.5	48.9	72.6	60.8	42.5	74.3	58.4
	1426	62.3	67.7	65.0	55.3	71.3	63.3	48.6	72.6	60.6
Ruled_Out	43	45.7	74.5	60.1	28.0	67.7	47.9	2.8	5.8	4.3
	134	59.5	73.2	66.3	49.5	76.2	62.9	46.0	79.7	62.8
	310	62.7	75.4	69.1	56.9	77.1	67.0	49.8	76.3	63.0

Table B.1 Hospital discharge results at error tolerance 0.20

Concept	Pos.		Min.Cover 2			Min.Cover 10			Min.Cover 20		
	Train		R	P	Avg	R	P	Avg	R	P	Avg
Present	1890	sem-1	42.5	63.6	53.0	33.5	73.6	53.5	28.2	78.5	53.4
		sem-2	61.9	65.6	63.8	59.8	72.0	65.9	57.3	74.2	65.8
Absent	2167	sem-1	72.3	77.3	74.8	66.1	80.4	73.3	59.6	82.0	70.8
		sem-2	80.0	78.9	79.5	77.5	81.6	79.5	75.7	83.0	79.3
Confirmed	1031	sem-1	59.0	67.9	63.5	50.7	72.4	61.6	48.3	73.5	60.9
		sem-2	74.8	69.1	71.9	73.5	72.4	73.0	72.2	73.9	73.1
Ruled_Out	211	sem-1	62.0	76.3	69.2	47.4	79.3	63.3	47.4	79.3	63.3
		sem-2	73.5	83.1	78.3	67.5	86.3	76.9	67.5	86.3	76.9

Table B.2 Hospital discharge at 50% training before (sem-1) and after (sem-2) semantic fine-tuning at error tolerance 0.20

APPENDIX C

SEMANTIC HIERARCHY FOR MANAGEMENT SUCCESSION

Class	Parent Class
Root_Class	
Event	Root_Class
Entity	Root_Class
Person	Entity
Person_Name	Person
Person_Title	Person
Generic_Person	Person
Organization	Entity
Org_Name	Organization
Org_Desig	Organization
Generic_Org	Organization
Position	Entity
Corporate_Post	Position
Generic_Role	Position
Name	Entity
Irrel_Org	Entity
Government	Irrel_Org
Location	Entity
Country	Location
City	Location
Region	Location
State	Location

Figure C.1 Semantic hierarchy for Management Succession (continued on next page)

Class	Parent Class
Artifact	Entity
Product	Artifact
Natural_Resource	Artifact
Technology	Artifact
Facility	Entity
Conceptual_Entity	Entity
Idea_or_Concept	Conceptual_Entity
Temporal_Concept	Idea_or_Concept
Date	Temporal_Concept
Absolute_Date	Date
Relative_Date	Date
Duration	Temporal_Concept
Time	Temporal_Concept
Spatial_Concept	Idea_or_Concept
Qualitative_Concept	Idea_or_Concept
Nationality	Qualitative_Concept
Quantitative_Concept	Idea_or_Concept
Number	Quantitative_Concept
Percentage	Number
Money	Quantitative_Concept
Communicate	Event
Hire	Event
Fire	Event
Keep_Job	Event
Leave_Job	Event
Former_or_Current	Event
Start	Event
Exist	Event
Past	Exist
Posess	Event
Create	Event
Purchase	Event
Hypothetical	Event

Figure C.2 Semantic hierarchy for Management Succession (continuation of previous figure)

APPENDIX D

SEMANTIC HIERARCHY FOR HOSPITAL DISCHARGE

Class	Parent Class
Root_Class	
Entity	Root_Class
Conceptual_Entity	Entity
Idea_or_Concept	Conceptual_Entity
Finding	Conceptual_Entity
Laboratory_or_Test_Result	Finding
Sign_or_Symptom	Finding
Temporal_Concept	Idea_or_Concept
Date	Temporal_Concept
Absolute_Date	Date
Relative_Date	Date
Duration	Temporal_Concept
Qualitative_Concept	Idea_or_Concept
Quantitative_Concept	Idea_or_Concept
Spatial_Concept	Idea_or_Concept
Body_Location_or_Region	Spatial_Concept
Body_Space_or_Junction	Spatial_Concept
Group	Conceptual_Entity
Professional_or_Occupational_Group	Group
Population_Group	Group
Patient_or_Disabled_Group	Group
Age_Group	Group
Family_Group	Group

Figure D.1 Semantic hierarchy for Hospital Discharge (continued on next page)

Class	Parent Class
Event	Root_Class
Phenomenon_or_Process	Event
Injury_or_Poisoning	Phenomenon_or_Process
Natural_Phenomenon_or_Process	Phenomenon_or_Process
Biologic_Function	Natural_Phenomenon_or_Process
Pathologic_Function	Biologic_Function
Cell_or_Molecular_Dysfunction	Pathologic_Function
Disease_or_Syndrome	Pathologic_Function
Mental_or_Behavioral_Dysfunction	Disease_or_Syndrome
Activity	Event
Behavior	Activity
Social_Behavior	Behavior
Individual_Behavior	Behavior
Daily_or_Recreational_Activity	Activity
Occupational_Activity	Activity
Health_Care_Activity	Occupational_Activity
Laboratory_Procedure	Health_Care_Activity
Therapeutic_or_Preventive_Procedure	Health_Care_Activity
Diagnostic_Procedure	Health_Care_Activity
Educational_Activity	Occupational_Activity
Governmental_or_Regulatory_Activity	Occupational_Activity
Research_Activity	Occupational_Activity
Molecular_Biology_Research_Technique	Research_Activity
Machine_Activity	Activity
Physiologic_Function	Biologic_Function
Cell_Function	Physiologic_Function
Molecular_Function	Physiologic_Function
Organism_Function	Physiologic_Function
Mental_Process	Organism_Function
Organ_or_Tissue_Function	Physiologic_Function
Genetic_Function	Molecular_Function
Experimental_Model_of_Disease	Pathologic_Function
Human_caused_Phenomenon_or_Process	Phenomenon_or_Process
Environmental_Effect_of_Humans	Human_caused_Phenomenon_or_Process

Figure D.2 Semantic hierarchy for Hospital Discharge (continued on next page)

Class	Parent Class
Physical_Object	Entity
Anatomical_Structure	Physical_Object
Embryonic_Structure	Anatomical_Structure
Acquired_Abnormality	Anatomical_Structure
Congenital_Abnormality	Anatomical_Structure
Fully_Formed_Anatomical_Structure	Anatomical_Structure
Body_Part_or_Organ	Fully_Formed_Anatomical_Structure
Cell_Component	Fully_Formed_Anatomical_Structure
Cell	Fully_Formed_Anatomical_Structure
Tissue	Fully_Formed_Anatomical_Structure
Macromolecular_Structure	Fully_Formed_Anatomical_Structure
Gene_or_Genome	Macromolecular_Structure
Organism	Physical_Object
Animal	Organism
Invertebrate	Animal
Vertebrate	Animal
Bird	Vertebrate
Amphibian	Vertebrate
Fish	Vertebrate
Mammal	Vertebrate
Human	Mammal
Reptile	Vertebrate
Bacterium	Organism
Plant	Organism
Alga	Plant
Alga	Plant
Fungus	Organism
Virus	Organism
Rickettsia_or_Chlamydia	Organism
Substance	Physical_Object
Body_Substance	Substance
Chemical	Substance
Food	Substance

Figure D.3 Semantic hierarchy for Hospital Discharge (continued on next page)

Class	Parent Class
Chemical_Viewed_Functionally	Chemical
Pharmacologic_Substance	Chemical_Viewed_Functionally
Indicator_or_Reagent	Chemical_Viewed_Functionally
Hazardous_or_Poisonous_Substance	Chemical_Viewed_Functionally
Biologically_Active_Substance	Chemical_Viewed_Functionally
Neuroreactive_Substance	Biologically_Active_Substance
Hormone	Biologically_Active_Substance
Enzyme	Biologically_Active_Substance
Vitamin	Biologically_Active_Substance
Prostaglandin	Biologically_Active_Substance
Immunologic_Factor	Biologically_Active_Substance
Chemical_Viewed_Structurally	Chemical
Inorganic_Chemical	Chemical_Viewed_Structurally
Element_or_Ion	Inorganic_Chemical
Isotope	Inorganic_Chemical
Inorganic_Compound	Inorganic_Chemical
Organic_Chemical	Chemical_Viewed_Structurally
Alkaloid	Organic_Chemical
Amino_Acid_or_Peptide_or_Protein	Organic_Chemical
Carbohydrate	Organic_Chemical
Eicosanoid	Organic_Chemical
Lactam	Organic_Chemical
Lipid	Organic_Chemical
Nucleic_Acid_or_Nucleotide	Organic_Chemical
Steroid	Organic_Chemical
Organophosphorus_Compound	Organic_Chemical
Biomedical_or_Dental_Material	Chemical_Viewed_Functionally
Manufactured_Object	Physical_Object
Medical_Device	Manufactured_Object
Research_Device	Manufactured_Object

Figure D.4 Semantic hierarchy for Hospital Discharge (continued on next page)

Class	Parent Class
Occupation_or_Discipline	Conceptual_Entity
Biomedical_Occupation_or_Discipline	Occupation_or_Discipline
Intellectual_Product	Conceptual_Entity
Classification	Intellectual_Product
Regulation_or_Law	Intellectual_Product
Amino_Acid_Sequence	Molecular_Sequence
Carbohydrate_Sequence	Molecular_Sequence
Nucleotide_Sequence	Molecular_Sequence
Group_Attribute	Conceptual_Entity
Functional_Concept	Idea_or_Concept
Body_System	Functional_Concept
Molecular_Sequence	Spatial_Concept
Geographic_Area	Spatial_Concept
Organism_Attribute	Conceptual_Entity
Language	Conceptual_Entity
Organization	Conceptual_Entity
Health_Care_Related_Organization	Organization
Health_Care_Related_Organization	Organization
Professional_Society	Organization
Self_help_or_Relief_Organization	Organization

Figure D.5 Semantic hierarchy for Hospital Discharge (continuation of previous figure)

Class	Parent Class
Symptom_Absent	Conceptual_Entity
Symptom_or_Diagnosis	Conceptual_Entity
Diagnosis	Pathologic_Function
Diagnosis_Preexisting	Diagnosis
Diagnosis_Past	Diagnosis
Symptom_Modifier	Idea_or_Concept
Symptom_Absent_Modifier	Symptom_Modifier
Symptom_Present_Modifier	Symptom_Modifier
Diagnosis_Modifier	Idea_or_Concept
Diagnosis_Preexisting_Modifier	Diagnosis_Modifier
Diagnosis_Past_Modifier	Diagnosis_Modifier
Diagnosis_or_Symptom	Biologic_Function
Questionable	Conceptual_Entity

Figure D.6 Semantic classes added for fine-tuned Hospital Discharge tagging

APPENDIX E

AUTOMATICALLY DERIVED CONCEPT DEFINITIONS FOR PERSON_IN, POSITION

These concept definitions were generated by CRYSTAL for the Management Succession concept *Person_In,Position*. Induction was done at error tolerance 0.20 from 479 randomly selected training documents. This is a complete list of definitions with coverage of at least five training instances.

Semantic classes have a prefix "ws_" rather than the angle brackets. The extracted syntactic constituent is marked with "==>". Coverage and errors on the training set (used by CRYSTAL's search control) is recorded for each concept definition. I have also included coverage and errors on a blind test set. The test set is one fourth the size of the training set.

The order of syntactic constituents is not necessarily the order that they are found in an instance. The current implementation of CRYSTAL does not include any ordering constraints on the syntactic constituents.

The two highest coverage definitions in this rule base (ID: 8833 and ID: 9130) extract *Person_In* from the subject and *Position* from the direct object. The verb is constrained to be "was named" for 8833 and the subject and object are virtually unconstrained. In 9130 the subject must have the semantic class <Person>, and the object must have the semantic class <Position>, but any passive verb including "was" will do.

CN-type Succession ID: 8833
Status: GENERALIZED
Constraints:
VERB::
mode: passive affirmative
root: name
terms: WAS NAMED
mod terms: WAS
head terms: NAMED
classes: ws_Past
mod class: ws_Past
head class: ws_Root_Class
OBJ:: ==> Position
mode: affirmative
classes: ws_Entity
mod class: ws_Root_Class
head class: ws_Root_Class
SUBJ:: ==> Person_In
mode: affirmative
classes: ws_Root_Class
mod class: ws_Root_Class
head class: ws_Root_Class
Coverage: 133, Errors: 25 (Coverage on test set: 36, Errors: 7)

CN-type Succession ID: 9130
Status: GENERALIZED
Constraints:
VERB::
mode: passive affirmative
terms: WAS
mod terms: WAS
classes: ws_Past
mod class: ws_Past
head class: ws_Root_Class
SUBJ:: ==> Person_In
mode: affirmative
classes: ws_Person
mod class: ws_Root_Class
head class: ws_Person
OBJ:: ==> Position
mode: affirmative
classes: ws_Position
mod class: ws_Root_Class
head class: ws_Root_Class
Coverage: 125, Errors: 10 (Coverage on test set: 40, Errors: 2)

CN-type Succession ID: 8849
Status: GENERALIZED
Constraints:
VERB::
mode: affirmative
root: name
terms: NAMED
head terms: NAMED
classes: ws_Root_Class
mod class: ws_Root_Class
head class: ws_Root_Class
SUBJ:: ==> Person_In
mode: affirmative
classes: ws_Person
mod class: ws_Root_Class
head class: ws_Person
OBJ:: ==> Position
mode: affirmative
classes: ws_Position
mod class: ws_Root_Class
head class: ws_Entity
Coverage: 119, Errors: 11 (Coverage on test set: 34, Errors: 2)

CN-type Succession ID: 8795
Status: GENERALIZED
Constraints:
OBJ:: ==> Position
mode: affirmative
classes: ws_Position
mod class: ws_Root_Class
head class: ws_Position
VERB::
mode: affirmative
classes: ws_Root_Class
mod class: ws_Root_Class
head class: ws_Root_Class
SUBJ:: ==> Person_In
mode: affirmative
terms: %COMMA%
mod terms: %COMMA%
classes: ws_Person_Name, ws_Corporate_Post
mod class: ws_Root_Class
head class: ws_Person_Name
Coverage: 61, Errors: 5 (Coverage on test set: 22, Errors: 3)

CN-type Succession ID: 8749
Status: GENERALIZED
Constraints:
OBJ:: ==> Position
mode: affirmative
classes: ws_Entity
mod class: ws_Root_Class
head class: ws_Entity
VERB::
mode: passive affirmative
terms: WAS
mod terms: WAS
classes: ws_Past
mod class: ws_Past
head class: ws_Root_Class
SUBJ:: ==> Person_In
mode: affirmative
classes: ws_Idea_or_Concept, ws_Person_Name
mod class: ws_Root_Class
head class: ws_Person_Name
Coverage: 35, Errors: 1 (Coverage on test set: 10, Errors: 1)

CN-type Succession ID: 8818
Status: GENERALIZED
Constraints:
SUBJ:: ==> Person_In
mode: affirmative
classes: ws_Person_Name
mod class: ws_Root_Class
head class: ws_Person_Name
OBJ:: ==> Position
mode: affirmative
classes: ws_Entity
mod class: ws_Root_Class
head class: ws_Root_Class
VERB::
mode: passive affirmative
terms: WAS
mod terms: WAS
classes: ws_Past
mod class: ws_Past
head class: ws_Root_Class
REL-VERB::
mode: affirmative
classes: ws_Event
mod class: ws_Root_Class
head class: ws_Root_Class
Coverage: 35, Errors: 7 (Coverage on test set: 6, Errors: 1)

.

CN-type Succession ID: 8984
Status: GENERALIZED
Constraints:
SUBJ:: ==> Person_In
mode: affirmative
classes: ws_Person_Name
mod class: ws_Root_Class
head class: ws_Person_Name
VERB::
mode: passive affirmative
terms: WAS
mod terms: WAS
classes: ws_Past
mod class: ws_Past
head class: ws_Root_Class
PP:: ==> Position
mode: affirmative
classes: ws_Corporate_Post
mod class: ws_Root_Class
head class: ws_Root_Class
Coverage: 31, Errors: 6 (Coverage on test set: 6, Errors: 2)

CN-type Succession ID: 9180
Status: GENERALIZED
Constraints:
SUBJ:: ==> Person_In
mode: affirmative
classes: ws_Person
mod class: ws_Root_Class
head class: ws_Person
PP:: ==> Position
mode: affirmative
classes: ws_Corporate_Post
mod class: ws_Root_Class
head class: ws_Corporate_Post
VERB::
mode: affirmative
terms: WAS
mod terms: WAS
classes: ws_Past
mod class: ws_Past
head class: ws_Root_Class
Coverage: 31, Errors: 5 (Coverage on test set: 7, Errors: 2)

CN-type Succession ID: 8918
Status: GENERALIZED
Constraints:
VERB::
mode: active affirmative
root: become
classes: ws_Start
mod class: ws_Root_Class
head class: ws_Start
SUBJ:: ==> Person_In
mode: affirmative
classes: ws_Person
mod class: ws_Root_Class
head class: ws_Person
OBJ:: ==> Position
mode: affirmative
classes: ws_Corporate_Post
mod class: ws_Root_Class
head class: ws_Entity
Coverage: 30, Errors: 5 (Coverage on test set: 7, Errors: 0)

CN-type Succession ID: 8946
Status: GENERALIZED
Constraints:
SUBJ:: ==> Person_In
mode: affirmative
classes: ws_Person_Name
mod class: ws_Root_Class
head class: ws_Person_Name
VERB::
mode: affirmative
classes: ws_Root_Class
mod class: ws_Root_Class
head class: ws_Root_Class
PP:: ==> Position
prep: TO
mode: affirmative
terms: TO
classes: ws_Position
mod class: ws_Root_Class
head class: ws_Position
Coverage: 28, Errors: 5 (Coverage on test set: 7, Errors: 3)

CN-type Succession ID: 8771
Status: GENERALIZED
Constraints:
VERB::
mode: passive affirmative
root: name
terms: WAS NAMED
mod terms: WAS
head terms: NAMED
classes: ws_Past
mod class: ws_Past
head class: ws_Root_Class
SUBJ:: ==> Person_In
mode: affirmative
terms: %COMMA%
mod terms: %COMMA%
classes: ws_Person_Name, ws_Corporate_Post
mod class: ws_Root_Class
head class: ws_Person_Name, ws_Corporate_Post
OBJ:: ==> Position
mode: affirmative
classes: ws_Position
mod class: ws_Root_Class
head class: ws_Position
Coverage: 26, Errors: 0 (Coverage on test set: 9, Errors: 0)

CN-type Succession ID: 9038
Status: GENERALIZED
Constraints:
VERB::
mode: active affirmative
classes: ws_Root_Class
mod class: ws_Root_Class
head class: ws_Root_Class
SUBJ::
mode: affirmative
classes: ws_Entity
mod class: ws_Root_Class
head class: ws_Entity
PP:: ==> Position
mode: affirmative
classes: ws_Corporate_Post
mod class: ws_Root_Class
head class: ws_Corporate_Post
OBJ:: ==> Person_In
mode: affirmative
terms: %COMMA%
mod terms: %COMMA%
classes: ws_Person_Name
mod class: ws_Root_Class
head class: ws_Person_Name
Coverage: 15, Errors: 2 (Coverage on test set: 5, Errors: 4)

CN-type Succession ID: 8873
Status: GENERALIZED
Constraints:
SUBJ:: ==> Person_In
mode: affirmative
classes: ws_Person_Name
mod class: ws_Root_Class
head class: ws_Person_Name
VERB::
mode: affirmative
root: appoint
terms: APPOINTED
head terms: APPOINTED
classes: ws_Event
mod class: ws_Root_Class
head class: ws_Event
OBJ:: ==> Position
mode: affirmative
classes: ws_Corporate_Post
mod class: ws_Root_Class
head class: ws_Corporate_Post
Coverage: 15, Errors: 3 (Coverage on test set: 5, Errors: 0)

CN-type Succession ID: 8757
Status: GENERALIZED
Constraints:
PP:: ==> Position
prep: AS
mode: affirmative
terms: AS
classes: ws_Corporate_Post
mod class: ws_Root_Class
head class: ws_Corporate_Post
VERB::
mode: active affirmative
root: succeed
classes: ws_Root_Class
mod class: ws_Root_Class
head class: ws_Root_Class
OBJ::
mode: affirmative
classes: ws_Person
mod class: ws_Root_Class
head class: ws_Person
SUBJ:: ==> Person_In
mode: affirmative
classes: ws_Person_Name
mod class: ws_Root_Class
head class: ws_Entity
Coverage: 12, Errors: 0 (Coverage on test set: 6, Errors: 1)

CN-type Succession ID: 9072
Status: GENERALIZED
Constraints:
PP:: ==> Position
prep: AS
mode: affirmative
terms: AS
classes: ws_Corporate_Post
mod class: ws_Root_Class
head class: ws_Entity
SUBJ:: ==> Person_In
mode: affirmative
classes: ws_Person_Name
mod class: ws_Root_Class
head class: ws_Person_Name
VERB::
mode: active affirmative
root: succeed
classes: ws_Root_Class
mod class: ws_Root_Class
head class: ws_Root_Class
Coverage: 12, Errors: 0 (Coverage on test set: 6, Errors: 1)

CN-type Succession ID: 8912
Status: GENERALIZED
Constraints:
VERB::
mode: active affirmative
root: name
terms: NAMED
mod terms: <null>
head terms: NAMED
classes: ws_Root_Class
mod class: ws_Root_Class
head class: ws_Root_Class
SUBJ::
mode: affirmative
classes: ws_Org_Name
mod class: ws_Root_Class
head class: ws_Org_Name
OBJ:: ==> Position, Person_In
mode: affirmative
classes: ws_Person, ws_Corporate_Post
mod class: ws_Entity
head class: ws_Corporate_Post
Coverage: 12, Errors: 2 (Coverage on test set: 1, Errors: 0)

CN-type Succession ID: 9027
Status: GENERALIZED
Constraints:
VERB::
mode: active affirmative
classes: ws_Root_Class
mod class: ws_Root_Class
head class: ws_Root_Class
OBJ:: ==> Position, Person_In
mode: affirmative
classes: ws_Person, ws_Corporate_Post
mod class: ws_Person
head class: ws_Corporate_Post
SUBJ::
mode: affirmative
classes: ws_Organization
mod class: ws_Root_Class
head class: ws_Organization
Coverage: 12, Errors: 2 (Coverage on test set: 1, Errors: 0)

CN-type Succession ID: 8858
Status: GENERALIZED
Constraints:
VERB::
mode: active affirmative
root: name
terms: NAMED
head terms: NAMED
classes: ws_Root_Class
mod class: ws_Root_Class
head class: ws_Root_Class
SUBJ::
mode: affirmative
classes: ws_Entity
mod class: ws_Root_Class
head class: ws_Entity
OBJ:: ==> Position, Person_In
mode: affirmative
terms: OF
mod terms: OF
classes: ws_Person_Name, ws_Corporate_Post
mod class: ws_Entity
head class: ws_Corporate_Post
Coverage: 11, Errors: 2 (Coverage on test set: 0, Errors: 0)

CN-type Succession ID: 9104
Status: GENERALIZED
Constraints:
VERB::
mode: affirmative
root: appoint
terms: APPOINTED
head terms: APPOINTED
classes: ws_Event
mod class: ws_Root_Class
head class: ws_Event
SUBJ:: ==> Person_In
mode: affirmative
classes: ws_Entity
mod class: ws_Root_Class
head class: ws_Entity
OBJ:: ==> Position
mode: affirmative
terms: OF
mod terms: OF
classes: ws_Corporate_Post
mod class: ws_Root_Class
head class: ws_Corporate_Post
Coverage: 11, Errors: 2 (Coverage on test set: 6, Errors: 0)

CN-type Succession ID: 9010
Status: GENERALIZED
Constraints:
SUBJ:: ==> Person_In
mode: affirmative
terms: %COMMA%
mod terms: %COMMA%
classes: ws_Person_Name
mod class: ws_Root_Class
head class: ws_Person_Name
VERB::
mode: active affirmative
terms: WILL
mod terms: WILL
classes: ws_Root_Class
mod class: ws_Root_Class
head class: ws_Root_Class
PP:: ==> Position
prep: AS
mode: affirmative
terms: AS
classes: ws_Corporate_Post
mod class: ws_Root_Class
head class: ws_Entity
Coverage: 11, Errors: 2 (Coverage on test set: 3, Errors: 2)

CN-type Succession ID: 9052
Status: GENERALIZED
Constraints:
VERB::
mode: active affirmative
mod terms: <null>
classes: ws_Root_Class
mod class: ws_Root_Class
head class: ws_Root_Class
OBJ:: ==> Person_In
mode: affirmative
classes: ws_Entity
mod class: ws_Root_Class
head class: ws_Entity
SUBJ::
mode: affirmative
classes: ws_Entity
mod class: ws_Root_Class
head class: ws_Entity
PP:: ==> Position
mode: affirmative
terms: THE OF
mod terms: THE OF
classes: ws_Generic_Role
mod class: ws_Root_Class
head class: ws_Generic_Role
Coverage: 10, Errors: 1 (Coverage on test set: 2, Errors: 0)

CN-type Succession ID: 9157
Status: GENERALIZED
Constraints:
SUBJ:: ==> Person_In
mode: affirmative
classes: ws_Person_Name
mod class: ws_Root_Class
head class: ws_Person_Name
OBJ:: ==> Position
mode: affirmative
terms: THE OF
mod terms: THE OF
classes: ws_Entity
mod class: ws_Root_Class
head class: ws_Entity
VERB::
mode: active affirmative
root: assume
classes: ws_Root_Class
mod class: ws_Root_Class
head class: ws_Root_Class
Coverage: 10, Errors: 2 (Coverage on test set: 2, Errors: 0)

.

CN-type Succession ID: 8781
Status: GENERALIZED
Constraints:
SUBJ::
mode: affirmative
classes: ws_Entity
mod class: ws_Root_Class
head class: ws_Entity
VERB::
mode: active affirmative
mod terms: <null>
classes: ws_Root_Class
mod class: ws_Root_Class
head class: ws_Root_Class
OBJ:: ==> Position, Person_In
mode: affirmative
classes: ws_Corporate_Post
mod class: ws_Root_Class
head class: ws_Corporate_Post
REL-VERB::
mode: affirmative
terms: %COMMA% %PERIOD%
mod terms: %COMMA%
head terms: %PERIOD%
classes: ws_Entity
mod class: ws_Root_Class
head class: ws_Root_Class
Coverage: 8, Errors: 1 (Coverage on test set: 1, Errors: 1)

CN-type Succession ID: 8802
Status: GENERALIZED
Constraints:
VERB::
mode: active affirmative
mod terms: <null>
classes: ws_Root_Class
mod class: ws_Root_Class
head class: ws_Root_Class
OBJ:: ==> Person_In
mode: affirmative
terms: %COMMA%
mod terms: %COMMA%
classes: ws_Person_Name, ws_Corporate_Post
mod class: ws_Root_Class
head class: ws_Person_Name
PP:: ==> Position
mode: affirmative
terms: THE
mod terms: THE
classes: ws_Position
mod class: ws_Root_Class
head class: ws_Position
Coverage: 7, Errors: 0 (Coverage on test set: 0, Errors: 0)

CN-type Succession ID: 9002
Status: GENERALIZED
Constraints:
VERB::
mode: active affirmative
mod terms: <null>
classes: ws_Root_Class
mod class: ws_Root_Class
head class: ws_Root_Class
OBJ:: ==> Person_In
mode: affirmative
classes: ws_Person_Name
mod class: ws_Root_Class
head class: ws_Person_Name
SUBJ::
mode: affirmative
classes: ws_Organization
mod class: ws_Root_Class
head class: ws_Entity
REL-VERB:: ==> Position
mode: affirmative
terms: TO
mod terms: TO
classes: ws_Corporate_Post
mod class: ws_Root_Class
head class: ws_Root_Class
Coverage: 7, Errors: 1 (Coverage on test set: 0, Errors: 0)

CN-type Succession ID: 9078
Status: GENERALIZED
Constraints:
OBJ:: ==> Person_In
mode: affirmative
classes: ws_Person_Name
mod class: ws_Root_Class
head class: ws_Person_Name
VERB::
mode: active affirmative
root: succeed
classes: ws_Root_Class
mod class: ws_Root_Class
head class: ws_Root_Class
REL-OBJ:: ==> Position
mode: affirmative
terms: WHO WAS
mod terms: WHO WAS
classes: ws_Corporate_Post, ws_Past
mod class: ws_Past
head class: ws_Root_Class
SUBJ::
mode: affirmative
classes: ws_Person
mod class: ws_Root_Class
head class: ws_Person
Coverage: 7, Errors: 1 (Coverage on test set: 1, Errors: 1)

CN-type Succession ID: 9111
Status: GENERALIZED
Constraints:
PP:: ==> Position
prep: AS
mode: affirmative
terms: AS
classes: ws_Corporate_Post
mod class: ws_Root_Class
head class: ws_Corporate_Post
VERB::
mode: active affirmative
classes: ws_Event
mod class: ws_Root_Class
head class: ws_Event
OBJ:: ==> Person_In
mode: affirmative
classes: ws_Person_Name
mod class: ws_Root_Class
head class: ws_Person_Name
Coverage: 7, Errors: 1 (Coverage on test set: 2, Errors: 1)

CN-type Succession ID: 8969
Status: GENERALIZED
Constraints:
VERB::
mode: active affirmative
mod terms: <null>
classes: ws_Root_Class
mod class: ws_Root_Class
head class: ws_Root_Class
SUBJ::
mode: affirmative
classes: ws_Organization
mod class: ws_Root_Class
head class: ws_Entity
REL-VERB:: ==> Position
mode: affirmative
terms: TO %PERIOD%
mod terms: TO
head terms: %PERIOD%
classes: ws_Corporate_Post
mod class: ws_Entity
head class: ws_Root_Class
OBJ:: ==> Person_In
mode: affirmative
classes: ws_Person_Name
mod class: ws_Root_Class
head class: ws_Person_Name
Coverage: 6, Errors: 1 (Coverage on test set: 0, Errors: 0)

CN-type Succession ID: 9200
Status: GENERALIZED
Constraints:
SUBJ::
mode: affirmative
classes: ws_Entity
mod class: ws_Root_Class
head class: ws_Entity
VERB::
mode: active affirmative
classes: ws_Root_Class
mod class: ws_Root_Class
head class: ws_Root_Class
OBJ:: ==> Person_In
mode: affirmative
classes: ws_Person_Name
mod class: ws_Root_Class
head class: ws_Person_Name
REL-OBJ:: ==> Position
mode: affirmative
terms: WHO WAS
mod terms: WHO WAS
classes: ws_Entity, ws_Past
mod class: ws_Past
head class: ws_Entity
Coverage: 6, Errors: 1 (Coverage on test set: 1, Errors: 1)

CN-type Succession ID: 8894
Status: GENERALIZED
Constraints:
VERB::
mode: active affirmative
classes: ws_Event
mod class: ws_Root_Class
head class: ws_Event
OBJ:: ==> Person_In
mode: affirmative
classes: ws_Person_Name
mod class: ws_Root_Class
head class: ws_Person_Name
SUBJ::
mode: affirmative
classes: ws_Entity
mod class: ws_Root_Class
head class: ws_Entity
PP:: ==> Position
prep: AS
mode: affirmative
terms: AS
classes: ws_Corporate_Post
mod class: ws_Root_Class
head class: ws_Corporate_Post
Coverage: 5, Errors: 0 (Coverage on test set: 1, Errors: 1)

CN-type Succession ID: 8975
Status: GENERALIZED
Constraints:
OBJ:: ==> Person_In
mode: affirmative
classes: ws_Corporate_Post
mod class: ws_Root_Class
head class: ws_Entity
REL-VERB:: ==> Position
mode: affirmative
terms: %PERIOD%
head terms: %PERIOD%
classes: ws_Corporate_Post
mod class: ws_Corporate_Post
head class: ws_Root_Class
VERB::
mode: active affirmative
mod terms: <null>
classes: ws_Root_Class
mod class: ws_Root_Class
head class: ws_Root_Class
SUBJ::
mode: affirmative
classes: ws_Organization
mod class: ws_Root_Class
head class: ws_Entity
Coverage: 5, Errors: 0 (Coverage on test set: 0, Errors: 0)

CN-type Succession ID: 9088
Status: GENERALIZED
Constraints:
SUBJ:: ==> Person_In
mode: affirmative
classes: ws_Person_Name
mod class: ws_Root_Class
head class: ws_Person_Name
VERB::
mode: active affirmative
root: be
terms: BE
head terms: BE
classes: ws_Exist
mod class: ws_Root_Class
head class: ws_Exist
OBJ:: ==> Position
mode: affirmative
classes: ws_Corporate_Post
mod class: ws_Root_Class
head class: ws_Corporate_Post
Coverage: 5, Errors: 0 (Coverage on test set: 4, Errors: 1)

CN-type Succession ID: 8933
Status: GENERALIZED
Constraints:
SUBJ::
mode: affirmative
classes: ws_Entity
mod class: ws_Root_Class
head class: ws_Entity
VERB::
mode: active affirmative
classes: ws_Root_Class
mod class: ws_Root_Class
head class: ws_Root_Class
OBJ:: ==> Position, Person_In
mode: affirmative
terms: THE
mod terms: THE
classes: ws_Person_Name, ws_Generic_Role
mod class: ws_Entity
head class: ws_Entity
Coverage: 5, Errors: 1 (Coverage on test set: 0, Errors: 0)

CN-type Succession ID: 9062
Status: GENERALIZED
Constraints:
VERB::
mode: active affirmative
root: serve
terms: WILL SERVE
mod terms: WILL
head terms: SERVE
classes: ws_Root_Class
mod class: ws_Root_Class
head class: ws_Root_Class
SUBJ:: ==> Person_In
mode: affirmative
classes: ws_Person
mod class: ws_Root_Class
head class: ws_Person
PP:: ==> Position
prep: AS
mode: affirmative
terms: AS
classes: ws_Corporate_Post
mod class: ws_Root_Class
head class: ws_Corporate_Post
Coverage: 5, Errors: 1 (Coverage on test set: 1, Errors: 1)

Bibliography

[Aha *et al.* 1991] Aha, D., Kilber, D., Albert, M. Instance-Based Learning Algorithms. *Machine Learning*, 6, 37-66, 1991.

[Amit *et al.* 1995] Yamit, A., Geman, D., Wilder, K. Recognizing Shapes from Simple Queries about Geometry. Technical Report, Department of Statistics, University of Chicago, Department of Mathematics and Statistics, University of Massachusetts, 1995.

[Breiman *et al.* 1984] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. Classification and Regression Trees. Wadsworth Statistic/Probability Series, 1984.

[Brill 1994] Brill, E. Some Advances in Transformation-Based Part of Speech Tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 722-727, 1994.

[Cardie 1993] Cardie, C. A Case-Based Approach to Knowledge Acquisition for Domain-Specific Sentence Analysis. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, 798-803, 1993.

[Charniak 1995] Charniak, E. Parsing with Context-free Grammars and Word Statistics, Technical Report CS-95-28, Department of Computer Science, Brown University, 1995.

[Church 1988] Church, K. A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. In *Proceedings of the Second Conference on Applied Natural Language Processing*, 136-143, 1988.

[Church *et al.* 1991] Church, K., Gale, W., Hanks, P., Hindle, D. Using Statistics in Lexical Analysis. *Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon.* Lawrence Erlbaum Associates, Publishers, 115-164, 1991.

[Clark and Niblett 1989] Clark, P. and Niblett, T. The CN2 Induction Algorithm. *Machine Learning*, 3, 261-283, 1989.

[Cost and Salzberg 1993] Cost, S. and Salzberg, S. A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features. *Machine Learning*, 10, 57-78, 1993.

[Fisher and Riloff 1992] Fisher, D. and Riloff, E. Applying Statistical Methods to Small Corpora: Benefitting from a Limited Domain. In *Working Notes of 1992 AAAI Fall Symposium Series: Probabilistic Approaches to Natural Language*, 1992.

[Fisher *et al.* 1995] Fisher, D., Soderland, S., McCarthy, J., Feng, F., Lehnert, W. Description of the UMass System as Used for MUC-6. In *Proceedings of the Sixth Message Understanding Conference*, Morgan Kaufmann Publishers, 221-236, 1995.

[Huffman 1996] Huffman, S. Learning Information Extraction Patterns from Examples. *Connectionist, Statistical, and Symbolic approaches to Learning for Natural Language Processing.* Springer, 246-260, 1996.

[Kim and Moldovan 1992] Kim, J. and Moldovan, D. PALKA: A System for Linguistic Knowledge Acquisition. Technical Report PKPL 92-8, USC Department of Electrical Engineering Systems, 1992.

[Krupka 1995] Krupka, G. Description of the SRA System as Used for MUC-6. In *Proceedings of the Sixth Message Understanding Conference*, Morgan Kaufmann Publishers, 221-236, 1995.

[Lehnert *et al.* 1983] Lehnert, W., Dyer M., Johnson P., Yang C.J., Harley S. BORIS – An Experiment in In-Depth Understanding of Narratives. *Artificial Intelligence*, 20, 15-62, 1983.

[Lindberg *et al.* 1993] Lindberg, D., Humphreys, B., McCray, A. Unified Medical Language Systems. *Methods of Information in Medicine*, 32(4), 281-291, 1993.

[Magerman 1995] Magerman, D. Statistical Decision-Tree Models for Parsing. In *Proceedings of the 33rd Annual Meeting of the ACL*, 1995.

[Michalski 1983] Michalski, R. S. and Chilausky, R. L. Learning by Being Told and Learning from Examples. *Policy Analysis and Information Systems*, 219-244, 1980.

[Michalski 1983] Michalski, R. S. A Theory and Methodology of Inductive Learning. *Artificial Intelligence*, 20, 111-161, 1983.

[Miller *et al.* 1990] Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D. and Miller, K. J. Introduction to WordNet: An On-line Lexical Data Base. In the *Journal of Lexicography*, vol 3, no. 4, pp 235-244, 1990.

[Mingers 1989] Mingers, J. An Empirical Comparison of Selection Measures for Decision-Tree Induction. *Machine Learning*, 3, 319-342, 1989.

[Mitchell 1978] Mitchell, T. Version Spaces: an Approach to Concept Learning. PhD thesis, Department of Electrical Engineering, Stanford University, 1978.

[Mitchell 1982] Mitchell, T. Generalization as search. *Artificial Intelligence*, 18, 203-226, 1982.

[MUC-3 1991] *Proceedings of the Third Message Understanding Conference*, Morgan Kaufmann Publishers, 1991.

[MUC-4 1992] *Proceedings of the Fourth Message Understanding Conference*, Morgan Kaufmann Publishers, 1992.

[MUC-5 1993] *Proceedings of the Fifth Message Understanding Conference*, Morgan Kaufmann Publishers, 1993.

[MUC-6 1995] *Proceedings of the Sixth Message Understanding Conference*, Morgan Kaufmann Publishers, 1995.

[Murthy *et al.* 1994] Murthy, S.K., Kasif, S., and Salzberg, S. A System for Induction of Oblique Decision Trees. *Journal of Artificial Intelligence Research*, 2, 1-32, 1994.

[Pagallo and Haussler 1990] Pagallo, G. and Haussler, D. Boolean Feature Discovery in Empirical Learning. *Machine Learning*, 5, 71-99, 1990.

[Quinlan 1986] Quinlan, J.R. Induction of Decision Trees. *Machine Learning*, 81-106, 1986.

[Quinlan 1993] Quinlan, J.R. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.

[Quinlan and Cameron-Jones 1995] Quinlan, J.R. and Cameron-Jones, R.M. Oversearching and Layered Search in Emperical Learning. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 1019-1024, 1995.

[Riloff 1993] Riloff, E. Automatically Constructing a Dictionary for Information Extraction Tasks. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, 811-816, 1993.

[Riloff 1996] Riloff, E. Automatically Generating Extraction Patterns from Untagged Text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 1044-1049, 1996.

[Rivest 1987] Rivest, R. Learning Decision Lists. *Machine Learning*, 2, 1987.

[Soderland and Lehnert 1994] Soderland, S. and Lehnert, W. Wrap-Up: a Trainable Discourse Module for Information Extraction. *Journal of Artificial Intelligence Research*, 2, 131-158, 1994.

[Soderland *et al.* 1995] Soderland, S., Fisher, D., Aseltine, J., Lehnert, W. CRYSTAL: Inducing a Conceptual Dictionary. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 1314-1321, 1995.

[Sundheim 1992] Sundheim, B. Overview of the Fourth Message Understanding Evaluation and Conference. In *Proceedings of the Fourth Message Understanding Conference*, Morgan Kaufmann Publishers, 3-21, 1992.

[Vere 1975] Vere, S. Induction of Concepts in the Predicate Calculus. In *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, 281-287, 1975.

[Vere 1980] Vere, S. Multilevel Counterfactuals for Generalizations of Relational Concepts and Productions. *Artificial Intelligence*, 14, 139-164, 1980.

[Will 1993] Will, C. Comparing Human and Machine Performance for Natural Language Information Extraction: Results for English Microelectronics from the MUC-5 Evaluation. In *Proceedings of the Fifth Message Understanding Conference*, Morgan Kaufmann Publishers, 53-68, 1993.

[Yarowsky 1992] Yarowsky, D. Word Sense Disambiguation Using Statistical Models of Roget's Categories Trained on Large Corpora. In *Proceedings of the Fourteenth International Conference on Computational Linguistics*, 454-460, 1992.