CRYSTAL: Learning Domain-specific Text Analysis Rules

Stephen G. Soderland

Natural Language Processing Laboratory
University of Massachusetts at Amherst

November 6, 1996

Abstract

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 predefined set of concepts in a specific 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 and out of corporate management positions. 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. Each domain needs a set of text extraction rules based on the vocabulary, semantic classes, and writing style peculiar to the domain and the target concept.

This paper 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 training size.

CRYSTAL belongs to the class of machine learning algorithms called covering algorithms, and presents a novel control strategy with time and space complexity independent of the feature size. 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 extraction. While simple rules are adequate to capture the most salient regularities in the training data, the subtlety and variability of unrestricted natural language require rich expressiveness in the rules for high performance.

Contents

Chapter 1

Introduction

Developing automated text understanding systems is of practical as well as theoretical interest. The vast quantity of text data available on-line can best be handled by intelligent agents whose understanding of the text goes beyond keyword search. This requires a system that can reliably extract both the explicitly stated information and that which can be reasonably inferred.

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 to a manageable size has been that of *information extraction* (IE). In an IE system, the task is restricted to identifying a predefined set of concepts in a specific domain and ignoring other information. A series of Message Understanding Conferences [MUC-3 1991, MUC-4 1992, MUC-5 1993, MUC-6 1995] sponsored by ARPA has given impetus to this approach.

I will present CRYSTAL [Soderland *et al.* 1995], a system that automatically learns domain-specific rules for information extraction. Learning extraction 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 or out of top corporate management positions. The only information considered relevant are 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

```
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

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

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 extraction rules to identify references to relevant information. Rules that apply to the text in Figure 1.1 might look for patterns such as:

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

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

Text extraction rules produce a fragmentary view of the text. For example, a rule based on the 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*.

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 1.3: Information extracted using pattern 1

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

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 1.4: Information extracted using pattern 2

An IE system needs to consolidate the output of the text extraction 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 of empty slots.

This paper will concentrate on the rules that extract information based on local context, keeping in mind that this only one of many components in a full IE system. Text extraction rules form a critical information source for information extraction, but one which, unfortunately, is highly domain-specific and must be acquired for each new domain.

1.2 The Need for Trainable Systems

Text extraction 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 in which the relevant information is symptoms and diagnoses. Instead, rules 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 1.5: 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 extraction rules is a difficult and time-consuming task.

Building rules by hand requires both system expertise and domain expertise. A reasonable approach, and the paradigm adopted by the MUC evaluations, is corpus-based system development. Domain experts take a representative set of several hundred 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 extraction 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 to define the underlying 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 extraction rule. CRYSTAL tests each rule that it proposes against the entire training corpus to ensure that the rule has not been generalized too far.

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 extraction rules automatically.

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 are 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 government positions, partners in law firms, or appointment of union officials. Being a member of the board of directors is also not relevant, although chairmanship of a company is relevant.

The output of Management Succession 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 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 extraction rules that are based on local context

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)

cannot make such distinctions and must rely on later discourse processing to filter out incorrect extractions.

The corpus of Management Succession texts includes the 200 texts provided by the MUC-6 performance evaluation. 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 following statistics. 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

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 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.

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 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

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

phrases not marked as 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 inserting SGML tags around relevant phrases to label that phrase's role in a target concept. A point-and-click interface was used to facilitate the process.

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>).

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

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.

In domains with output represented as multi-slot case frames, the SGML tags 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.

The SGML tags that are added directly to the training texts 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 extraction 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.

What features of the local context will be sufficient for CRYSTAL's text extraction rules, and what representation of the text should be passed to CRYSTAL? 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 word constraints. 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 <Position>. 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 word constraints as well as se-

mantic constraints 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 class constraints and exact word constraints 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 extraction 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.

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. The only requirement is that the instances are presented as a flat list of syntactic constituents. 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 sub-classes, for example a constraint requiring the class <Person> covers the sub-classes <Person Name> and <Generic Person> in Management Succession instances.

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.

A concept definition has a list of constraints that operate on syntactic constituents of an instance. For example, one concept definition may have constraints on the subject, verb, and direct object, while another definition

may have constraints on the subject and on three prepositional phrases. Each of these syntactic constituents may have any of the following constraints.

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

Figure 3.1: Constraints in a concept definition

Any combination of the above constraints may be included in a concept definition. The term constraint is an unordered list of words that must be included in the syntactic constituent. The classes constraint is an unordered list of semantic classes that must be present, either directly or through an IS-A relationship. Head terms and classes are those found as the last term of the phrase, or just before punctuation, before a preposition, or before an adverb. All other terms are considered modifiers.

The mode constraint is used for "affirmative" or "negative", "active" or "passive". CRYSTAL attaches no meaning to these labels, and will accept whatever modes are assigned by the sentence analyzer. The root constraint is used if the sentence analyzer provides the verb root. The preposition constraint requires that a prepositional phrase have a particular preposition.

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 of the instance. For example, a concept definition for *Succession Event* may fill the *Person_In* slot with the subject and the *Person_Out* slot with the direct object of the instance.

A few concrete examples of concept definitions will make this clear.

3.2 A Few Sample Concept Definitions

Figure 3.2 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
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 Office>
Mode: affirmative

Figure 3.2: 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. Words not found in the semantic lexicon for the domain are given the class <Root Class> (not shown in the figure).

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 3.3 show a concept definition that applies to the instance in Figure 3.2. 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 3.3: 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 instantiates a case frame with *Person_In* filled by "He" and *Person_Out* filled by "Jack Harper, a company founder".

Figure 3.4 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 Office> in the relative clause. The relative clause must also include the terms "who" and "named".

After applying this concept definition to the instance in Figure 3.2, 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 include the desired information, but also have extraneous words that must be trimmed away by later processing in a full information extraction system.

3.3 Finding the Right Level of Generalization

Each of these concept definitions operates properly on the sample instance, but so would a large range of possible concept definitions. The definition in

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

Figure 3.4: A concept definition that applies to "Jack Harper, a company founder who was named chairman"

Figure 3.3 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 requires that there be no modifier term. Figure 3.5 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 3.5: An alternate concept definition for "He succeeds Jack Harper, a company founder"

The definition in Figure 3.5 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 3.5 is a better concept definition than that in Figure 3.3 or vice versa. This example is merely 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 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 text, 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-defined error tolerance.

CRYSTAL begins by selecting a positive instance of the target concept as a seed. CRYSTAL then takes the most specific definition that covers this instance and generalizes it as far as possible. The generalized definition is added to the rule base and another seed is selected from positive instances not yet covered by the rules. 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 from each positive instance of the concept being learned. These initial definitions are the most restrictive definition that covers the instance, requiring every term and every semantic class 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*, each subset of slots is

treated as a separate concept to be learned. 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 contains 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 has been hand-tagged with a succession event containing "He" as *Person_In* and "Jack Harper" as *Person_Out*. See Figure 2.4 for the tagged sentence and Figure 3.2 for the syntactic analysis that forms the basis for a training instance..

This initial concept definition will correctly extract *Person_In* and *Person_Out* from this sentence, but is so restrictive that it will never 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's task is to 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 word, as is %comma%). There are 32 possible ways to relax this constraint by dropping a subset of the terms. The total number of combinations that drop subsets of terms, head terms, and modifier terms is even greater.

Class constraints may be relaxed by moving up in the semantic hierarchy. For example <Generic Person> may be relaxed to <Person>, then to <Entity>, then to <Root Class>, which is equivalent to no constraint. The head class, modifier class, and class constraints for the direct object of this instance may be relaxed in over one hundred distinct ways.

With over one thousand possible ways 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 quickly drop constraints that are merely accidental features of the motivating instance and to retain those constraints that are essential to the target concept.

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 Office>
Head classes include:	<Corporate Office>
Modifier classes include:	<Past>, <Event>
Mode: affirmative	

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

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.

The key insight of CRYSTAL is to guide this relaxation by unifying the current definition with the most similar initial 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, I will use the terms "instance" and "initial concept definition" interchangeably.

CRYSTAL unifies a concept definition with a similar initial definition by dropping constraints that are not shared by the two definitions. 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 with <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 quickly dropped. 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 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, which covers two training instances with no errors. The next proposed generalization in part C covers four instances with no errors. In part D of Figure 4.2, relaxing constraints enough to cover the nearest positive instance will also cover two negative instances.

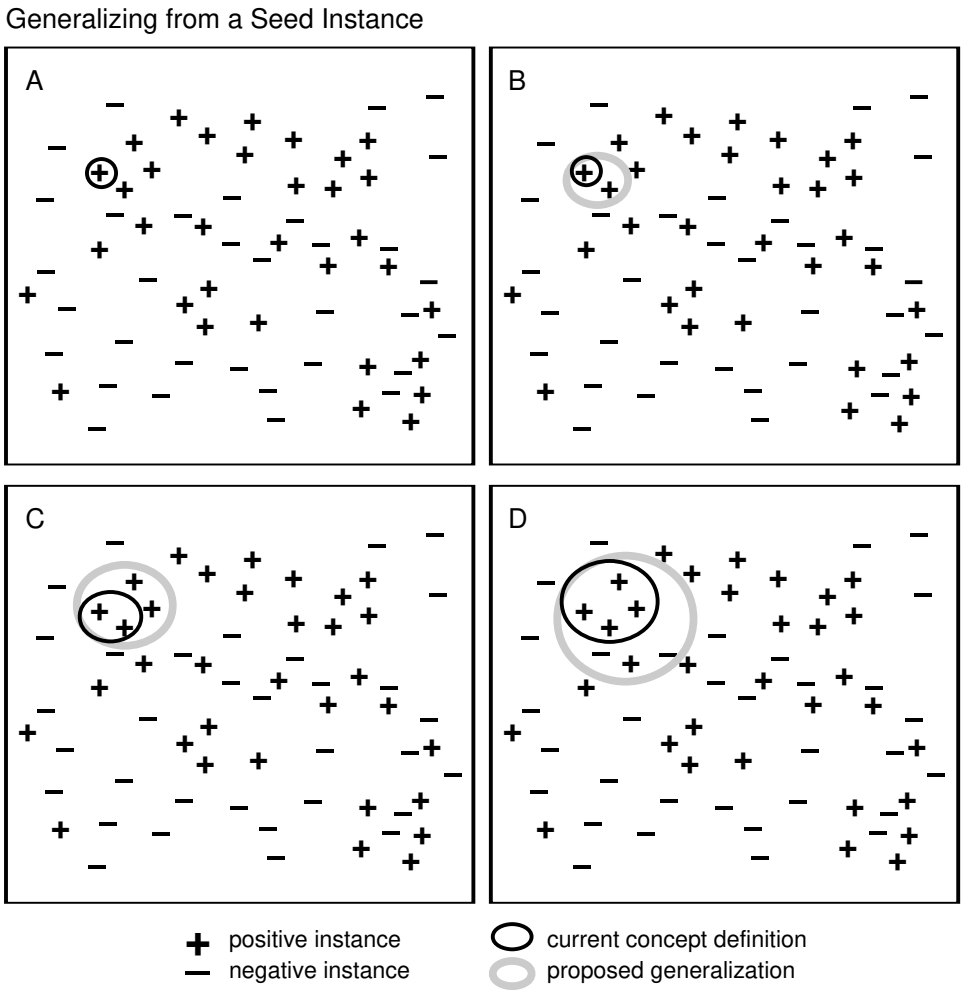

Figure 4.2: Each generalization step relaxes constraints just enough to cover the nearest positive instance.

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, and add the current definition (which covers four instances) to the rule base.

4.3 Finding Similar Concept Definitions

The distance metric used by CRYSTAL counts the number of relaxations of the current definition that would be needed to unify it with a similar definition.

Dropping one word from a term constraint counts as a single relaxation, as does dropping a mode or root constraint. Moving up in the semantic hierarchy counts as one relaxation for each level of the hierarchy. Entirely dropping a class constraint is equivalent to moving up to the root class, then dropping the class constraint.

Only concept definitions that extract the same slot values from the same constituents are candidates for similar 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 one definition has two prepositional phrases "as president" and "in 1983" and the other definition has one prepositional phrase "as chairman". CRYSTAL matches the most similar prepositional phrases, and leaves "in 1983" unmatched.

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

CRYSTAL includes a mechanism for biasing the distance function to give a greater weight to certain constraints and to certain syntactic constituents than to others. 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. This is discussed more fully in Appendix B.

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. As I will discuss later, the exact distance function is not critical to CRYSTAL's performance.

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, selects as a seed an initial definition not covered by the rules, and then finds a generalized definition based on that seed. This is repeated until all initial definitions have either been selected as seeds or are covered by the rules.

The main loop that generalizes a definition D, relaxes the constraints on D by finding a similar initial definition D' and creating the unification U of

```
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 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
```

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 hand-annotated training texts and 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. A certain amount of noise tolerance is desirable when processing unrestricted text.

One source of noise is the human annotators, who may overlook a phrase 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.

CRYSTAL is robust in another way also. There is a built-in redundancy in the concept definitions induced by CRYSTAL that tend to overcome sub-optimal choices made during the induction. Once a choice has been made, 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 seed. With any reasonable distance metric, one of these remaining seeds is likely to result in the high coverage definition. In that case the sub-optimal definition becomes redundant and causes no harm.

Even if CRYSTAL never finds a single definition to cover all 100 instances, this region in instance space will be covered by several overlapping definitions. These smaller coverage definitions, when taken together, may actually give better performance than a single high-coverage definition. Experiments reported in Section 5.4 show the result of increasing the amount of search for optimal definitions. This produces more a more compact set of high-coverage rules, but does not necessarily improve performance.

4.6 Time and Space Complexity of CRYSTAL

One of CRYSTAL's main contributions is a search control strategy that navigates a huge search space efficiently. Not only does CRYSTAL avoid backtracking, but its time and space requirements are independent of the size of

the feature sets. This is important in natural language processing, where the number of word-based features can be extremely large.

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 $O(pn)$.

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 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. This gives CRYSTAL an important time and space advantage over algorithms that must consider features exhaustively.

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 a *Person_In* and a *Person_Out*.

Let us begin with the initial definition, already presented in Figure 4.1, which is derived from the "seed" shown in Figure 4.3. This definition extracts *Person_In* from the subject and *Person_Out* from the direct object of the sentence.

Both the seed and the most similar instance have an identical subject and verb. The direct objects of each include a <Person Name> and <Organization>. The direct object has an appositive in both cases. Each sentence ends in a relative clause attached to the direct object that contains the class <Corporate Office> and <Past>.

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: A seed and the most similar instance found (The motivating sentence is shown rather than the entire initial definition.)

Unifying the seed with this instance produces the generalized concept definition shown in Figure 4.4. 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 actually it represents 16 individual relaxations. Words have been dropped from term constraints, head term constraints, and modifier term constraints. Class constraints requiring <Generic Person> and <Generic Organization> have been relaxed.

If any fewer than these 16 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.

Since the definition in Figure 4.4 has error rate 0.00, 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 4.5. 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 Office>, 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 Office> constraint, the definition covers six training instances with one error, giving it an error rate of 0.167 on the

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 Office>
Head classes include:	<Corporate Office>
Modifier classes include:	<Past>
Mode: affirmative	

Covers 2 training instances with 0 errors

Figure 4.4: 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"

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 Office>
Head classes include:	<Corporate Office>
Mode: affirmative	

Covers 4 training instances with 0 errors

Figure 4.5: 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"

training data. If the error tolerance has been set at 0.00 CRYSTAL halts and adds the definition in Figure 4.5 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 4.6. 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 4.6: 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 4.6 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 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.

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

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

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

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 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. Except as noted, the results are averages of ten random partitions of the texts.

CRYSTAL used the training set 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.

Performance is measured in terms of *recall* and *precision*. Recall is the percentage of instances of the target concept that were correctly identified. Precision is the percentage of extractions made that were correct. Suppose for example that there are 500 instances of the target concept in the test set and that rules induced by CRYSTAL found 300 of them. This gives recall of 60%. Further suppose that the rules made 100 false extractions in addition to the 300 correct ones. This gives precision of 75%.

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.

When statistical significance is mentioned, a two-tailed, paired t-test is used with $p<0.05$.

5.2 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.

5.2.1 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 graph in Figure 5.1 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 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 *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 5.2 give an indication that CRYSTAL was not saturated by this level of training.

Recall and precision are shown for four representative Management Succession concepts[1] 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[2]. This has the biggest impact on the multi-

[1] A table with results for all fifteen combinations of slots at 20%, 40%, and 80% training is given in Appendix A.

[2] 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

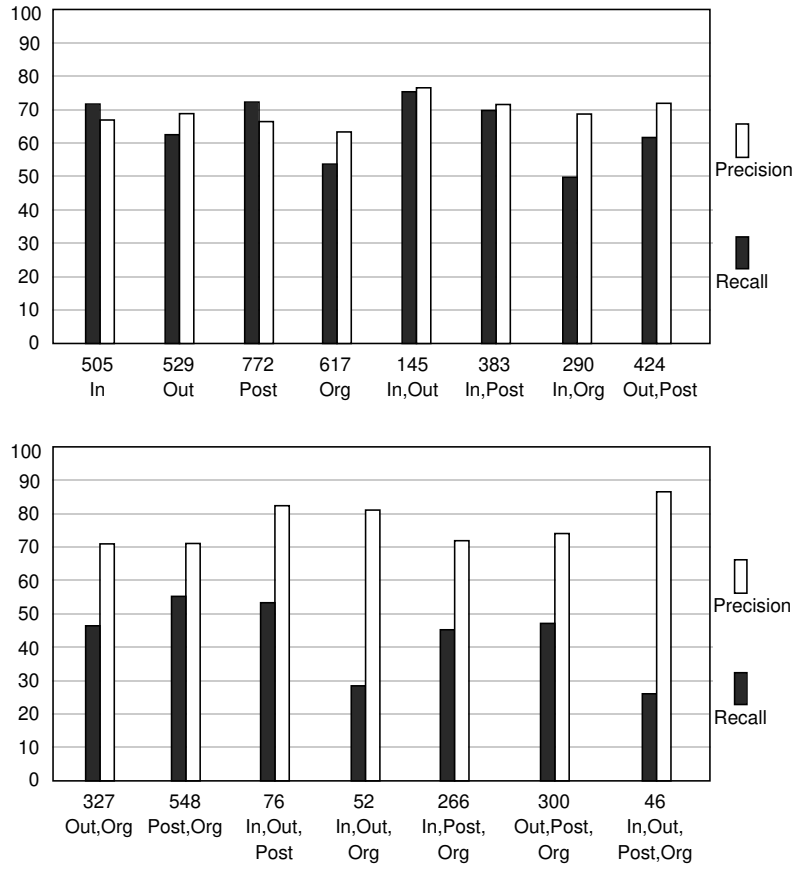

Figure 5.1: Management Succession performance: averages of 10 random partitions into 80% training, 20% test set

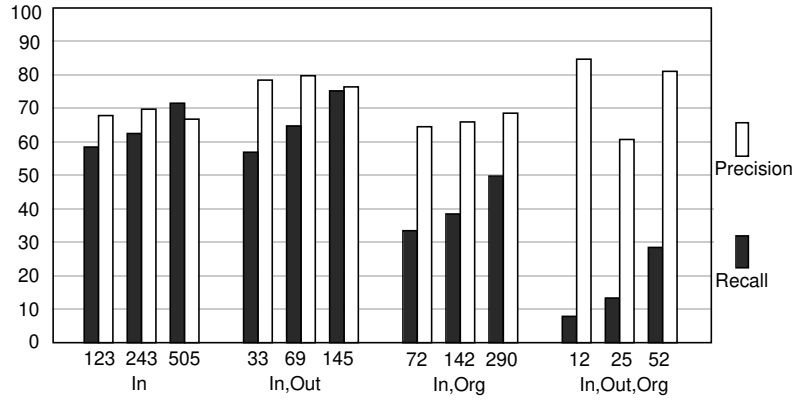

Figure 5.2: Management Succession learning curves at 20% training, 40% training, and 80% training

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 479 training texts (80% of the Management Succession corpus). These are broken down by how many training instances they cover. Note that the concepts with the highest performance are also the concepts with high coverage definitions (covering 50 or more training instances).

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

5.2.2 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 5.3 shows results for each of these concepts at three different training sizes. 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[3]. Results with fine-tuned semantic tagging will be presented later in this section.

in precision is significant. Only the fluctuation in precision of *In,Out,Org* is significant according to the t-test, but even this should be ignored due to the low sample size for this concept.]

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

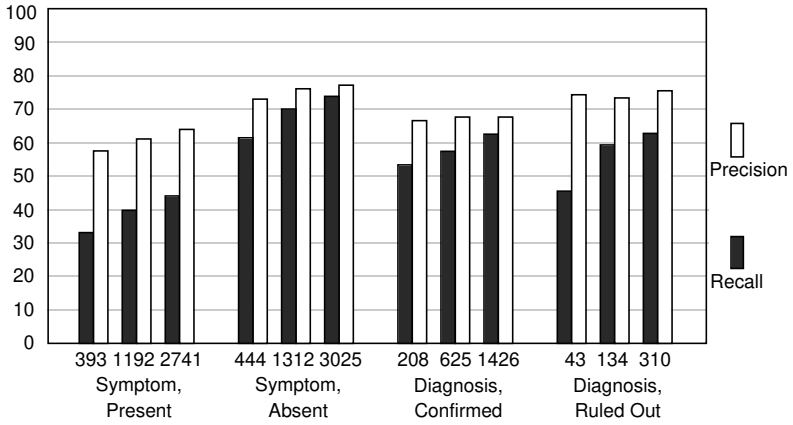

Figure 5.3: Hospital Discharge learning curves: averages of ten random partitions at 10% training, at 30% training, at 70% training

As in the Management Succession domain, recall increases with further training, while precision stays fairly flat[4]. It appears than 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 351 Hospital Discharge texts (70% training) is shown in Table 5.2.

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

A large proportion of the definitions cover less 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.

[4]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.

As the amount of training increases, a larger proportion of the recall comes from higher coverage definitions. Figure 5.4 shows performance for Hospital discharge rules when rules that cover less than ten training instances are discarded.

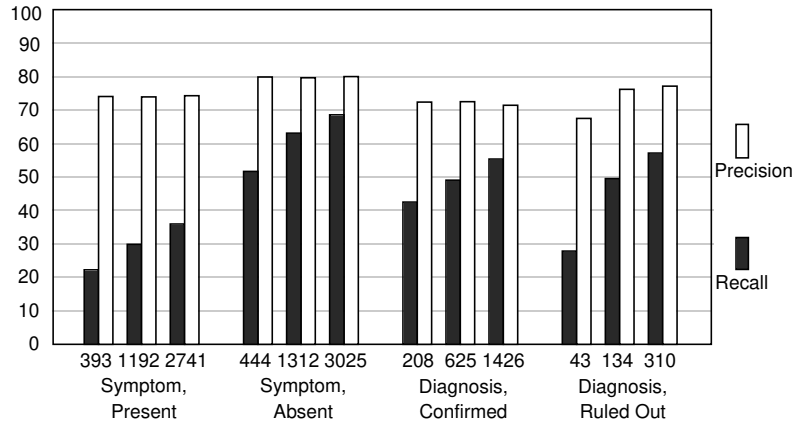

Figure 5.4: Only rules that cover ten or more training instances have been used in this Hospital Discharge learning curve.

With the low coverage definitions removed, precision is closer to the ideal of 80 that would be expected at error tolerance 0.20[5]. A comparison with Figure 5.3 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 *Diagnosis,Ruled_Out*, this is another indication that the level of training is approaching saturation. More training will increase performance only slightly.

5.2.3 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 gave CRYSTAL (or hand-coded rules, as will be discussed in the following section) a difficult job finding features that reliably characterize the target concepts. Noisy data is inevitable when dealing with automatic analysis of unrestricted text. The syntactic analysis or the semantic classes may be too coarse to make the necessary distinctions. Human annotators are liable to make errors or to differ among each other in creating the training.

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

One source of noise that I was able to partially 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 5.5. Recall and precision are now in the 60's to 80's for all four concepts, with *Symptom,Present* still lagging behind the others.

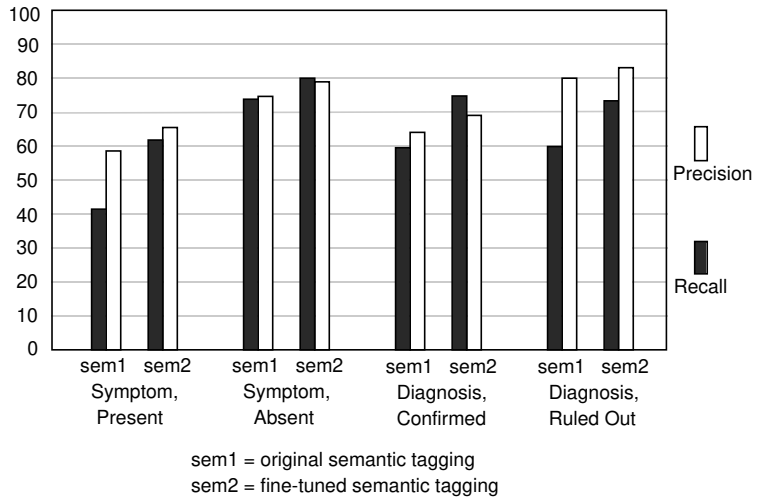

Figure 5.5: Hospital Discharge results with fine-tuned semantic tagging, at 50% training

These results are for a single partition of the corpus into 251 training texts and 251 test texts that were kept blind with respect to changes to the semantic

tagging. Note that 50% of the corpus is used as training, rather than the 70% training shown in Figure 5.3.

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

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 Management Succession after semantic tagging has been fine-tuned

A question remains for each of these domains. How close to optimal are CRYSTAL's results given the training set? One way to assess this is to compare the results with that of hand-coded rules.

5.3 Comparison with Hand-coded Rules

To find out how close CRYSTAL comes to optimal rules given the training set, 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 this domain for one and a half years. I also created rules by hand for three of the Management Succession concepts.

5.3.1 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.

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 Manage-

ment Succession were created without exceptions, both for CRYSTAL and hand-coded rules.

The methodology for creating rules by hand was 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 instances 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 definitions against the entire training set, listing the correct extractions and extraction errors for each definition.

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.

5.3.2 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 definitions. Before semantic fine-tuning, definitions 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 this is a training instance 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 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 an *Organization*

in a PP (prepositional phrase) and a *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 problems 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.

5.3.3 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 induction. The last column in the table uses the average of recall and precision to compute the ratio of CRYSTAL's performance to that of the hand-coded rules.

Before semantic fine-tuning:

Concept	CRYSTAL			Hand-coded			Ratio of
	R	P	Avg	R	P	Avg	Avg. R,P
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
	R	P	Avg	R	P	Avg	Avg. R,P
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

CRYSTAL achieves 93% the performance of hand-coded rules for three of the concepts and 88% as well 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. CRYSTAL achieves over 90% of the performance of hand-coded rules for these three concepts, with performance equal to hand-coded for *Person_In,Organization*.

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,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

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 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 less 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 close CRYSTAL comes to optimal performance is considered in the next chapter. If CRYSTAL expends more effort search-

ing for an optimal generalization from each seed, will this raise the over-all performance of the rule set?

5.4 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 size b and branching size k. When CRYSTAL generalizes from a seed instance, a *beam set* of size b is maintained. These are the best b generalized definitions found so far.

For each definition in the beam set, CRYSTAL finds k distinct relaxations by unifying with the most similar initial definition, the next most similar, and so forth. This produces a list of kb generalized definitions, which is sorted to keep the best b distinct definitions. The metric 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 $b = 1$. I ran experiments for the Management Succession domain and the Hospital Discharge domain at a range of beam sizes. Beam size was set to 1, 2, 5, and 10, with branching size equal to the beam size.

Figure 5.6 shows results at beam size 1, 2, 5, and 10 for four representative Management Succession concepts. These results are averages of ten random partitions with 40% of the corpus as training. The shaded dot indicates the average of recall and precision.

Figure 5.7 shows results of a similar beam search experiment for the Hospital Discharge domain. These are averages of ten random partitions with 50% of the corpus as training, no fine-tuning of the semantic tagging.

The total computation time[6] for all four Management Succession concepts averaged 14 minutes at $b = 1$, 18 minutes at $b = 2$, 36 minutes at $b = 5$, and 82 minutes at $b = 10$. Computation for the Hospital Discharge concepts took longer, since there were ten times as many positive instances, but a similar

[6] on a DEC ALPHA AXP 3000

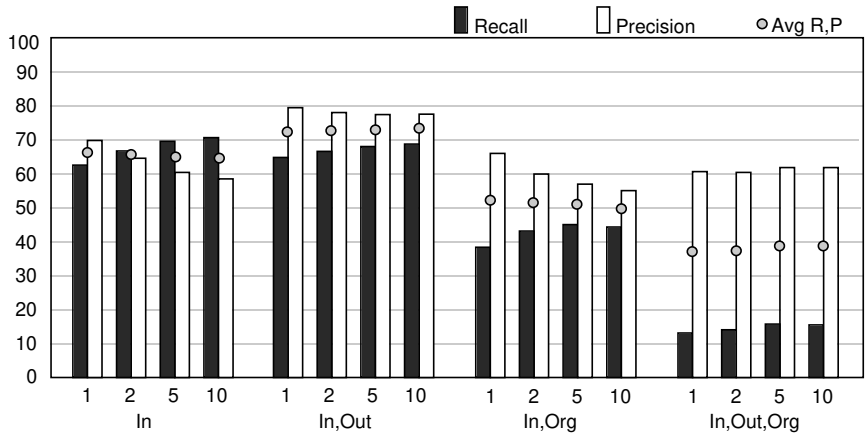

Figure 5.6: Management Succession results at beam size 1, 2, 5, and 10

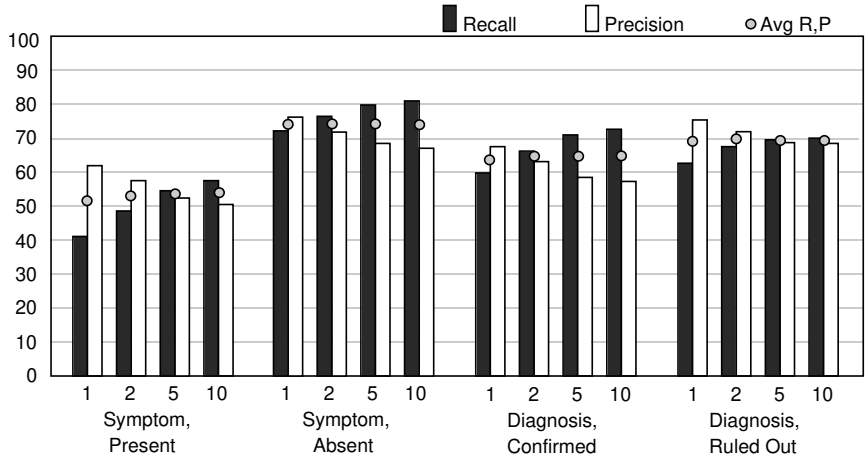

Figure 5.7: Hospital Discharge results at beam size 1, 2, 5, and 10

ratio held: 1.7 hours at $b = 1$, 2.8 hours at $b = 2$, 7.1 hours at $b = 5$, and 17.3 hours at $b = 10$.

Increasing the beam size 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 size 1 to beam size 2. There is little effect from moving from beam size 5 to 10. This holds for nearly all concepts in both domains. Most of the changes in recall and precision are statistically significant, and hardly any changes in average recall and precision as shown in Figure 5.8.

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.

	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 5.8: Statistical significance of changes in beam size

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

5.5 Manipulating a Recall-Precision Trade-off

A graphic illustration of the trade-off between recall and precision is shown in Figure 5.9. 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 recall. An alternate set of concept definitions could gain high recall at the expense of precision.

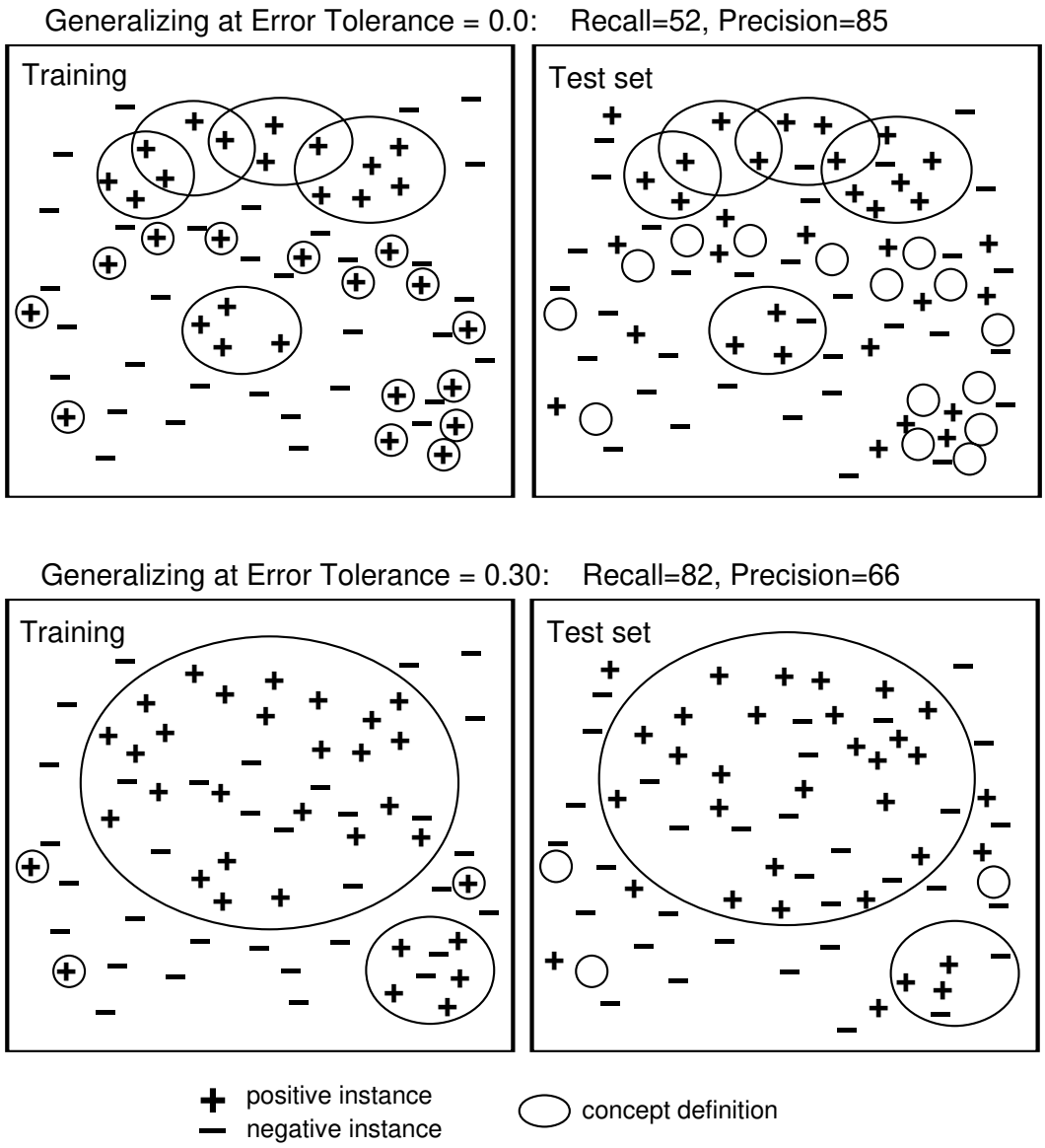

Figure 5.9: An instance space in which high recall or high precision is possible, but not both

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 5.9 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 5.9, 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 *Succession 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 deliberately manipulate the trade-off between recall and precision. 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.

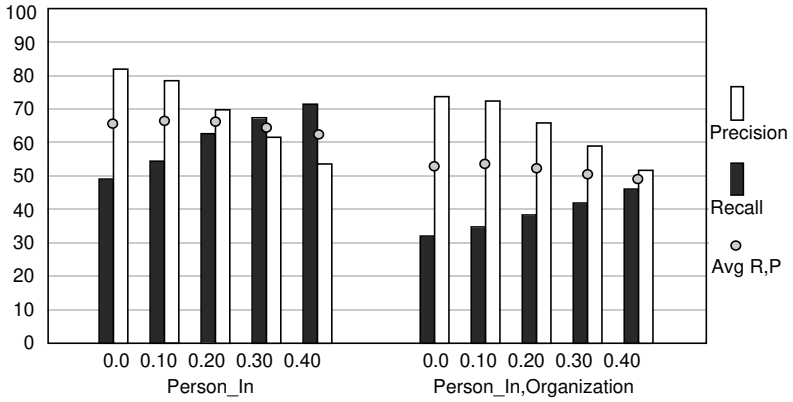

Figure 5.10: Management Succession results at error tolerance 0.0 to 0.40

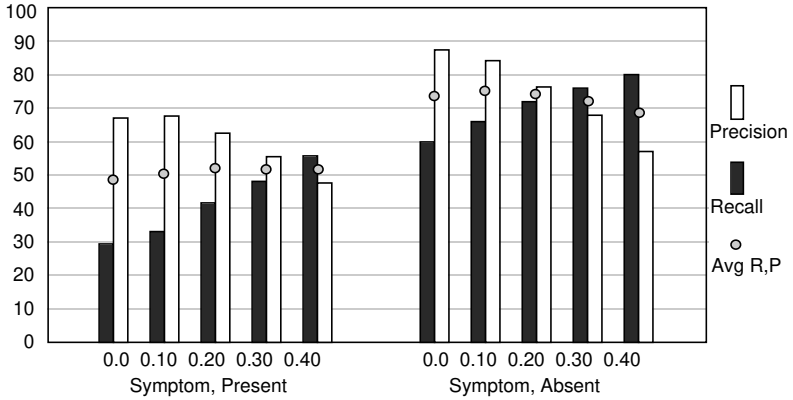

Figure 5.11: Hospital Discharge results at varying error tolerance

Figures 5.10 and 5.11 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[7].

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

[7]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 5.12 shows the interaction of error tolerance and min-coverage[8]. 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.

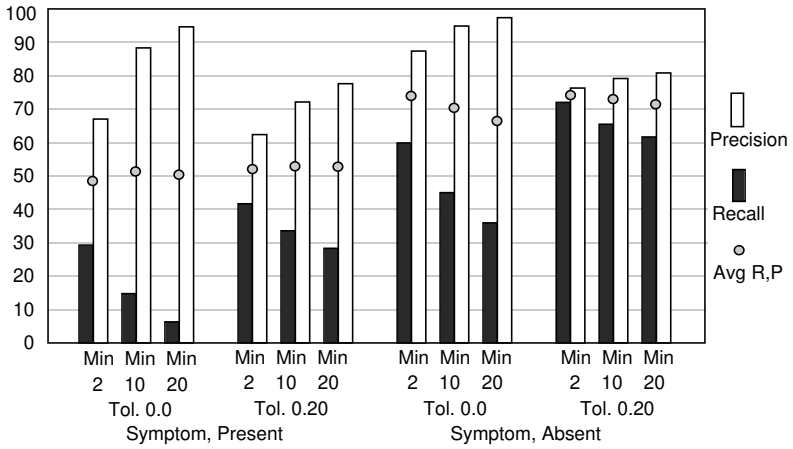

Figure 5.12: Effect of minimum coverage parameter at error tolerance 0.0 and 0.20

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.

5.6 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 seems to be consistent with this analysis.

Table 5.7 show 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. The CPU time shown includes the time required to read the

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

training instances and semantic hierarchy into memory and to write the rules generated to disk.

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

5.7 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 an optimal set of rules from a training 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, equalling 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 and 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 to merely 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 upon by more extensive search. The limit to CRYSTAL's performance comes rather 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, with diminishing returns eventually setting 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 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.

If high performance is at a premium, the most effective strategy is to customize the semantic tagging of individual words to the information extraction task. This has a greater impact than increasing training size. 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 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 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.

```
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*

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 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. Excep-

```
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

tions 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 halts generalization and discards the over-generalized definition. The previous version of the definition, which was within error tolerance, is added to the rules.

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 covers a region of instance space that includes 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 relatively small number of training instances compared to 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 is just as likely to exclude a positive as negative instances 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

```
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

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 have 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.

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 another $O(e^2)$ 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. Ten random partitions with 40% training and 60% blind test set were used for Management Succession, and 50% training 50% testing for Hospital Discharge.

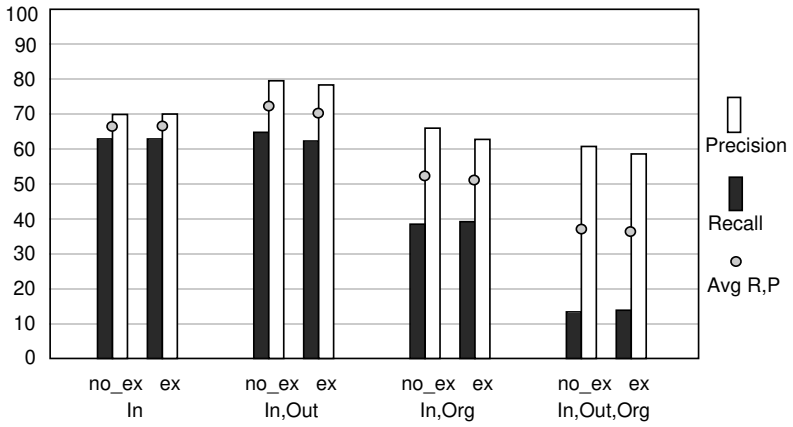

Figure 6.4: Comparison of Management Succession results with no exceptions and with exception

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

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.

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 has failed to improve performance. The next section presents experiments that restrict CRYSTAL's representation.

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

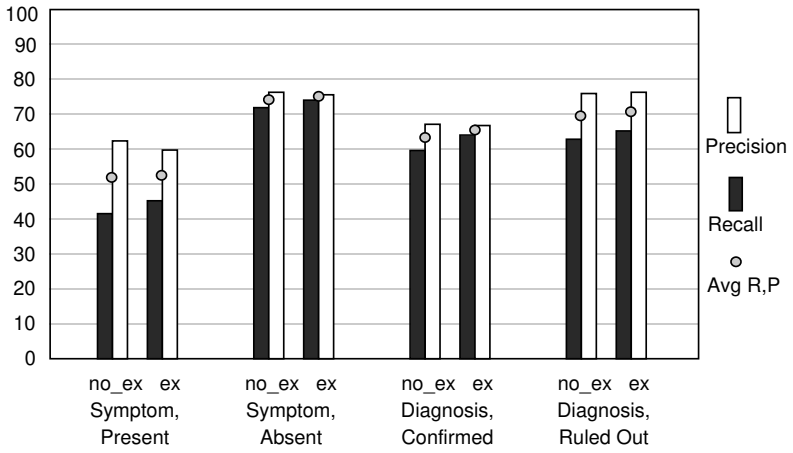

Figure 6.5: Comparison of Hospital Discharge results with no exceptions and with exception

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 extraction 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 only unified 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 extraction 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.

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.

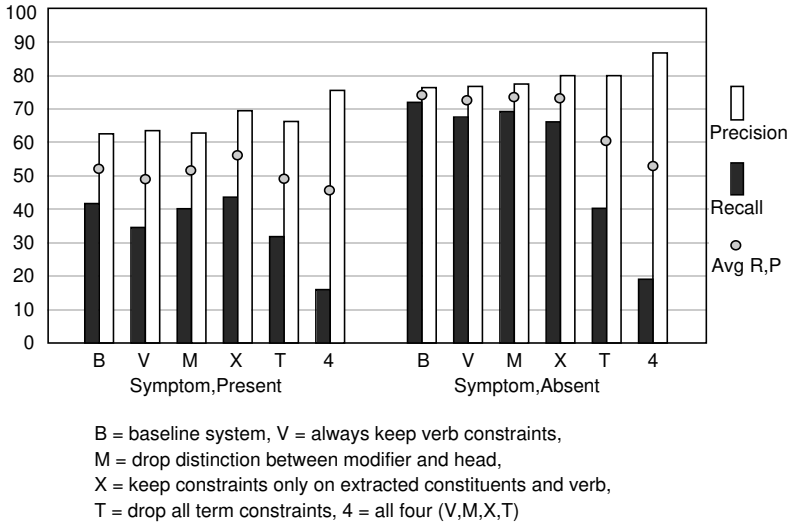

Figure 6.6: The effects of restricting CRYSTAL's representation in the Hospital Discharge domain

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

Single restrictions tend cause a moderate drop in recall and rise in precision[2]. Of the single restrictions, dropping terms hurt recall the worst. Applying all four restrictions invariably caused a large drop in recall.

Figure 6.7 shows results of these restrictions in two of the Management Succession concepts. The single-slot concept *Person_In* and the multi-slot concept *Person_In, Organization* are representative of results for this domain.

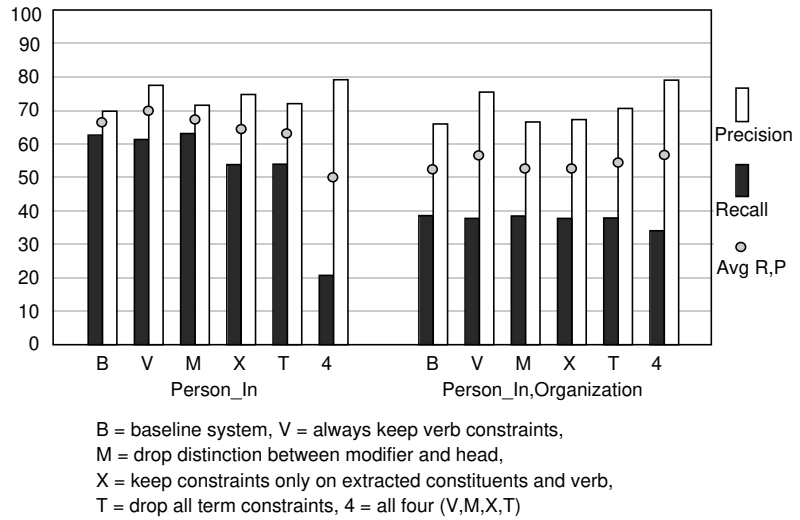

Figure 6.7: The effects of restricting CRYSTAL's representation in the Management Succession domain

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 drops only slightly from the baseline system to any of single restriction.

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

[2] All the differences from the baseline system for *Symptom, Present* are statistically significant but the increase in recall for "X", and the difference in precision and average precision and recall for "M". For *Symptom, Absent* all differences are significant but the difference in precision for "V".

[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".

plummets. The severely crippled representation is insufficient to distinguish many of the positive instances.

For some of the multi-slot concepts, the severely restricted representation was more nearly adequate. For three of the multi-slot 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 only "V" and "T" raised precision significantly[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 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 larger 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 5.3.2). 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.

[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".

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 for 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 systems 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 knowledge 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 1996] 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 1996]. 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 rules.

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 AutoSlog 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 [Kim and Moldovan 1992] system 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 version space algorithm (See Section 8.1.3) to generate text extraction 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."

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

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

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 8.2. 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.

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 tuneable 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

Succession

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

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 to replace or augment

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

the learned synonym lists.

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 extraction 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 restraints 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 a particular instance 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 extraction 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. First are the covering algorithms, the family of machine learning algorithms 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.

A reasonable question to ask, in view of the vast array of existing machine learning algorithms, is why invent CRYSTAL in the first place? The motivation for CRYSTAL came from demands of applying machine learning to natural language processing. The number of features used to describe natural language instances can become extremely large, particularly when some of the features are based on exact words.

Machine learning algorithms that use explicit feature vectors or have a basic step that considers all possible attributes can become impractical when the feature set becomes extremely large. I had an unsatisfactory experience using decision trees in a system called WRAP-UP [Soderland and Lehnert 1994] that learned to make inferences during discourse processing in information extraction. An excessive amount of memory and computation time was required to induce a decision tree when there were thousands of instances with thousands of features.

I wanted a representation that could easily combine syntactic, semantic, and lexical information. I also preferred an induction mechanism that mimics the process used in creating hand-coded text analysis rules, since this would help to make the resulting rules accessible to a human knowledge engineer. The covering algorithm methodology seemed the most natural approach.

8.1 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 will be Michalski's A^q algorithm, Clark and Niblett's CN2, and finally Mitchell's candidate elimination algorithm.

8.1.1 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 [89] also offer a readable explanation of A^q.

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 are 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. 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 [89] report using $s = 15$ for experiments with medical data sets[1].

The more significant difference in time complexity is that A^q takes time proportional to the number of attributes a. Each time a concept description is specialized, A^q considers all possible attributes in the feature set. This is not a big issue for data sets with at most a few dozen attributes that are often used as bench marks for machine learning algorithms[2]. In a natural language application with thousands of word-based attributes, this has a serious impact on efficiency.

An implementation of A^q could be written that considers only the attributes found in the seed instance rather than considering all attributes the feature set. This would limit A^q's concept definitions to positive attributes, but would give A^q a time complexity similar to a beam search version of CRYSTAL.

The next covering algorithm presented is CN2, which has time complexity similar to A^q and a similar top down approach.

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

[2]The UC Irvine collection of machine learning data sets

8.1.2 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 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. Minimizing entropy favors adding an attribute that comes closest to perfectly separating the classes into different partitions. 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 is as computationally expensive as it was for A^q. Each of the s concept descriptions in a star is specialized by adding any of $O(a)$ attributes, then each specialization is tested on all n training instances. Unlike A^q, there is no way to re-implement CN2 to avoid considering attributes exhaustively. If we treat the average number of attributes added to each description as a constant, each CN2 rule requires $O(asn)$ computations.

The entire decision list requires $O(asrn)$ computations, where r is the number of rules. The number of rules is bound by the n, giving a worst-case time complexity of $O(asn^2)$.

The next section describes the version space algorithm, which resembles covering algorithms, but has an even more exhaustive method of generating concept descriptions.

8.1.3 Version Space

The version space algorithm (or candidate elimination algorithm) of Tom Mitchell [82] identifies the space of *all* concept descriptions consistent with the training, defining this space with two sets. The set S lists the most specific concept descriptions consistent with the observed training instances, and the set G lists the most general descriptions. The version space algorithm updates each of these sets after each new training instance.

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. If concept descriptions can have any combination of a attributes, s and g have a worst case length of $O(a!)$.

This makes the version space algorithm impractical for any but small problems. An equally serious limitation is the version space algorithm's noise intolerance. Even a small amount of noise in the training data or insufficient attributes for classification causes the space of consistent descriptions to collapse to the empty set. While A^q or CN2 may be excessively slow for real-world problems involving large feature sets, version space is totally infeasible for such problems.

8.2 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 an 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 over all 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.

8.3 Decision Tree Algorithms

A well known family of machine learning algorithms is decision tree induction. 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.

8.3.1 C4.5

Decision tree algorithms have something in common with covering algorithms: top down processing and exhaustive consideration of attributes. Decision tree induction begins with an empty tree and recursively adds tests at each tree node to partition the instance space. The key step in building 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 makes the selection of a test even more expensive.

Many decision tree algorithms, including C4.5, base the feature selection metric on the information-theoretic measure, Shannon entropy.

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

where $(p_1, ...p_k)$ is a probability distribution among k classes at a given node. 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 is not critical to performance [Mingers 1989].

With its recursive partitioning, decision tree algorithms tend 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 recall and optimizing precision.

How well can decision trees handle extremely large number of attributes? The C4.5 algorithm must tabulate 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, giving computation time of $O(an)$ for each level of the tree and a total of $O(adn)$ computations, 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 increment a table based on a value found in a feature vector. CRYSTAL's basic operation is to test a proposed concept definition against a training instance. This 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. In the worst case, there could be a leaf for each training instance in a badly skewed tree with depth 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 $O(an)$.

8.3.2 GREEDY3

A variant of decision trees is the decision list, introduced by Ronald Rivest [87]. 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 at every node by using a multivariate test, a test that evaluates multiple features. This makes the decision list equivalent to a list of rules, each rule having several constraints.

GREEDY3 builds a decision list with a conjunct of Boolean literals as the test at each node. I will refer to these Boolean conjuncts as rules. Each time GREEDY3 adds a literal to the rule, it tabulates how often each possible literal is associated with a positive instance. The literal is chosen that has the highest probability of being found in a positive instance.

When enough literals have been added to the rule that it covers only positive instances, GREEDY3 adds the rule to the decision list. Instances covered by existing rules are removed from training set and GREEDY3 continues creating additional rules. When rules in the decision list cover all positive instances, 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 literal of 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 decision tree algorithms. It operates in a top down fashion, considers all Boolean literals exhaustively when selecting a literal to add to a node, and uses a selection metric based on how each literal performs on the training instances.

The computation required to select a new literal to add to a rule is $O(an)$,

where a is the number of literals (attributes) and n is the number to training instances. Let r be the number of rules in the decision list and k be the number of literals 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 $O(rn)$.

8.3.3 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 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. A separate C4.5 instance was created for each syntactic constituent of the CRYSTAL instance. These instances were presented to separate trees for each constituent: SUBJ, VERB, OBJ1, OBJ2, and PP[3].

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. The C4.5 instance included features that identified one of the syntactic constituents as the phrase to be classified. This encoding strategy works for single-slot extraction. Some other scheme would be needed for multi-slot concepts.

Boolean features were created for C4.5 from each constraint of each syntactic constituent of the CRYSTAL instance. The feature name began with the name of the constituent, followed by the name of the constraint and the actual term or class name.

For example an instance with "Chest x-ray" in the subject would include the features:

SUBJ-Terms-CHEST
SUBJ-Terms-X-RAY
SUBJ-Mod_Terms-CHEST
SUBJ-Head_Terms-X-RAY
SUBJ-Classes-Body_Location

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

SUBJ-Classes-Diagnostic_Procedure
SUBJ-Mod_Classes-Body_Location
SUBJ-Head_Classes-Diagnostic_Procedure

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[4].

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[5].

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,

[4]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.

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

CRYSTAL would faced the same problem of exceeding memory capacity if the Hospital Discharge corpus grew much larger.

8.4 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 tree or to concept descriptions in CRYSTAL and other covering algorithms.

Other differences are more fundamental to how the algorithm operates. Decision trees, decision lists, and the other covering algorithms all 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. CRYSTAL works from the opposite direction, beginning with maximally specific descriptions and generalizing by relaxing constraints.

This gives CRYSTAL a different "learning bias" than 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.

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 insufficient.

The greatest contrast between CRYSTAL and the other machine learning algorithms described here is in time complexity. 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.

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. 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. Training the instance based learning algorithms IB3 and PEBLS require $O(an^2)$ computations. Computation time for the version space algorithm is proportional to the square of the space requirement, which is $O(a!)$.

Some of these algorithms could be re-implemented to avoid considering possible attributes exhaustively. This is the case for A^q and for the IBL algorithms, but not for CN2, C4.5, GREEDY3, or version space. This would eliminate the a from time complexity and allow the algorithm to process much larger feature sets.

A re-implemented A^q would have time complexity similar to a beam search version of CRYSTAL. A re-implemented IBL algorithm would have time complexity of $O(n^2)$, only slightly greater than CRYSTAL's $O(rn)$.

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

Chapter 9

Conclusions

9.1 Combining Expressive Representation with an Efficient Learning Algorithm

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 feature set 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 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

[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.

irrelevant features. Including as many features as possible that *might* be useful increases CRYSTAL's performance and robustness.

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 b definitions found so far, where b is the beam size. A large beam size increases the amount of search effort and increases the likelihood that CRYSTAL will find the "best" generalized concept definition from a given seed instance.

A large beam size produces compact rule sets, but generally produces no improvement in average recall and precision on a blind test set. Increasing the beam size raises recall but lowers precision. The more extensive search results in concept definitions that seem reliable on the training set, but are 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 size of one.

9.2 Achieving High Performance with Modest Training Size

CRYSTAL's performance approaches 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 more difficult version of the same data set 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 compared 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 rep-

resentative 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 typically 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 will also require 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.

9.3 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.

9.3.1 Enhancements to CRYSTAL

CRYSTAL is a fairly mature system, as software goes, having been tested on several domains over the course of a year and a half. 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 some of the exceptions did not apply to 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.

9.3.2 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 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 "Mr. Lessin, whose resignation ...".

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 extraction 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 as shown in Figure 9.4.

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

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

will preserve ordering of the NP's and others will not. The distance between the two instances is presumably based on the best mapping.

This complication exists in the current CRYSTAL with multiple PP's (prepositional phrases) and has been solved efficiently as a bipartite matching problem. It is not clear that this solution will extend to a system with ordering constraints between the syntactic constituents.

9.4 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 learning a new set of text analysis rules to 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.

9.5 Acknowledgements

This material is based on work supported in part by the National Science Foundation, Library of Congress and Department of Commerce under cooperative agreement number EEC-9209623 and in part by NRaD Contract Number N66001-94-D-6054. Any opinions, findings and conclusions or recommendations expressed in this material are the author's and do not necessarily reflect those of the sponsor. Thanks also to David Aronow for help as a medical domain expert.

Bibliography

[Aha *et al.* 1991] Aha, D., Kilber, D., Albert, M. Instance-Based Learning Algorithms. *Machine Learning*, 6, 37-66, 1991.

[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. A Theory and Methodology of Inductive Learning. *Artificial Intelligence*, 20, 111-161, 1983.

[Mingers 1989] Mingers, J. An Empirical Comparison of Selection Measures for Decision-Tree Induction. *Machine Learning*, 3, 319-342, 1989.

[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 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 1996] 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.

[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.