INFORMATION EXTRACTION AS A BASIS FOR PORTABLE TEXT CLASSIFICATION SYSTEMS

A Dissertation Presented

by

ELLEN M. RILOFF

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

DOCTOR OF PHILOSOPHY

September 1994

Department of Computer Science

INFORMATION EXTRACTION AS A BASIS FOR PORTABLE TEXT CLASSIFICATION SYSTEMS

A Dissertation Presented

by

Ellen M. Riloff

Approved as to style and content by:

Wendy G. Lehnert, Chair

W. Bruce Croft, Member

Edwina L. Rissland, Member

Barbara H. Partee, Member

W. Richards Adrion, Department Head
Department of Computer Science

ACKNOWLEDGMENTS

First and foremost, I must thank my advisor, Wendy Lehnert, for her support through the years. I have always had the greatest respect for Wendy, but I am still amazed at her intelligence, energy, and creativity. Working with Wendy has been a privilege and I have learned an awful lot just by watching her do what she does best. I also thank Wendy for many personal insights and her tolerance with me through various crises, both perceived and real. Sometimes she seemed to know me better than I knew myself.

Bruce Croft has been a great asset to my committee and this dissertation has benefited enormously from his input. I thank Bruce for helping me navigate the mysterious world of information retrieval and for reassuring me that my work was interesting to people outside of the AI community. Edwina Rissland was extremely helpful in making sure that I kept one eye on the big picture and generality of my thesis. Edwina has been very supportive over the years and I am especially thankful for her advice and counsel during several decision points in my career.

I first met Barbara Partee in 1988 when she served as the second reader on my Master's project. That summer is one of my fondest memories of UMass, in large part because I had several meetings with Barbara that I enjoyed immensely. I thank Barbara for many insightful comments about my work and for stimulating conversations that always reminded me of why I got into this field in the first place.

Many friends have been instrumental in helping me survive the intellectual and psychological minefields of graduate school. I owe the deepest debt of gratitude to Kishore Swaminathan. Despite his propensity for stomach-wrenching puns, Kishore has been an invaluable source of support. I am not exaggerating when I say that I probably would not have survived graduate school without him. (Even though he thought I would never finish ... HA!) I thank Kishore for many spirited arguments, long phone conversations, and emotional support.

I have been extremely lucky to have Claire Cardie as a friend and colleague. Claire and I have been through a lot together and she has always been close by when I needed a friend. I am especially grateful to Claire for letting me ramble on, sometimes incoherently, when I needed an ear to lean on.

Copious thanks to David Fisher for his help with random things too numerous to mention and for proof-reading a draft of this thesis. Working with David has made me realize that I would truly enjoy having students of my own.

Special thanks to the other members of the NLP group: Joe McCarthy, Jon Peterson, and Stephen Soderland; David Skalak, for always bringing a smile to my face; Stefan Wermter, for fun traveling adventures; James Corbett, for sharing his wisdom on issues ranging from complexity theory to Dr. Seuss; Karen & Pete, for letting me escape to NH one week when I desperately needed to get away; and Priscilla Coe for handling many administrative tasks and helping me cope with the realities of everyday life. Priscilla is the best.

Also, a word of acknowledgment to the funding agencies that made this work possible: this research was supported by grants from NSF, ARPA, and ONR.

Last, but definitely not least, I thank my pet iguana, Kirbi, for helping me keep perspective on life and for introducing countless people to the wonderful world of reptiles.

ABSTRACT

INFORMATION EXTRACTION AS A BASIS FOR PORTABLE TEXT CLASSIFICATION SYSTEMS

SEPTEMBER 1994

ELLEN M. RILOFF

B.S., CARNEGIE MELLON UNIVERSITY

M.S., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Wendy G. Lehnert

Knowledge-based natural language processing systems have achieved good success with many tasks, but they often require many person-months of effort to build an appropriate knowledge base. As a result, they are not portable across domains. This knowledge-engineering bottleneck must be addressed before knowledge-based systems will be practical for real-world applications. This dissertation addresses the knowledge-engineering bottleneck for a natural language processing task called "information extraction". A system called AutoSlog is presented which automatically constructs dictionaries for information extraction, given an appropriate training corpus. In the domain of terrorism, AutoSlog created a dictionary using a training corpus and five person-hours of effort that achieved 98% of the performance of a hand-crafted dictionary that took approximately 1500 person-hours to build.

This dissertation also describes three algorithms that use information extraction to support high-precision text classification. As more information becomes available on-line, intelligent information retrieval will be crucial in order to navigate the information highway efficiently and effectively. The approach presented here represents a compromise between keyword-based techniques and in-depth natural language processing. The text classification algorithms classify texts with high accuracy by using an underlying information extraction system to represent linguistic phrases and contexts. Experiments in the terrorism domain suggest that increasing the amount of linguistic context can improve performance. Both AutoSlog and the text classification algorithms are evaluated in three domains: terrorism, joint ventures, and microelectronics. An important aspect of this dissertation is that AutoSlog and the text classification systems can be easily ported across domains.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

OVERVIEW

Manual knowledge engineering is not much fun. It is time-consuming, tedious, and difficult. Many graduate students in artificial intelligence spend a lot of time building knowledge bases by hand. Eventually, they spend a lot of time trying to avoid it.

Knowledge-based approaches hold great promise for achieving success with many problems in artificial intelligence. But knowledge-based techniques require a knowledge base. And it is not practical to manually build a new knowledge base for every problem.

The goal of this dissertation is to demonstrate that domain-specific knowledge for natural language processing can be acquired automatically from a training corpus. In other words, knowledge-based natural language processing systems do not always have to rely on manually encoded dictionaries and knowledge bases. From a practical perspective, this implies that knowledge-based natural language processing systems can be portable across domains with only minimal human effort.

This research addresses the issue of automated knowledge acquisition for natural language processing. We claim that knowledge-based natural language processing systems can be portable across domains. In Section 1.1, we describe a corpus-based approach to automated dictionary construction for domain-specific natural language processing, embodied in a system named AutoSlog. Section 1.2 presents a second theme of this dissertation: the integration of natural language processing with information retrieval. This section explains how natural language processing techniques can be applied to the problem of text classification. Section 1.3 outlines the big picture that we propose for developing *portable* knowledge-based systems for high-level language tasks. This section explains how AutoSlog can be combined with statistical text classification algorithms to produce a knowledge-based text classification system that can be easily ported across domains. Finally, Section 1.4 discusses the research contributions of this work, and Section 1.5 is a guide to the rest of the dissertation.

1.1 AutoSlog: A System for Automated Dictionary Construction

Over the years, knowledge-based natural language processing (NLP) systems have achieved good performance on many tasks (e.g., [Carbonell, 1979a, Cullingford, 1978, DeJong, 1982, Hayes and Weinstein, 1991, Lehnert, 1991, Mauldin, 1991]). However, most knowledge-based systems depend on a domain-specific knowledge base that was constructed by hand. Manual knowledge engineering is a problem for several reasons:

1. Manual knowledge engineering is time-intensive. It often requires many person-months of effort to build a knowledge base.

2. Manual knowledge engineering typically requires domain experts or experienced system developers. The human resources needed for knowledge engineering are often expensive and are not always readily available.

3. Manual knowledge engineering is difficult and prone to errors. Humans are not always the best judges of what types of knowledge are useful for a task. Humans can overlook knowledge that would be helpful, or can incorrectly judge knowledge to be useful.

4. Domain-specific systems that rely on a manually constructed knowledge base are not portable across domains. To apply a knowledge-based system to a new domain, the entire knowledge engineering process must be repeated.

Consequently, manual knowledge engineering is a major bottleneck for most knowledge-based systems. Knowledge-based systems will not be practical for real-world applications until this knowledge-engineering bottleneck is addressed. Most companies and potential users of this technology simply cannot afford to invest the time and financial resources that are necessary to build knowledge bases by hand.

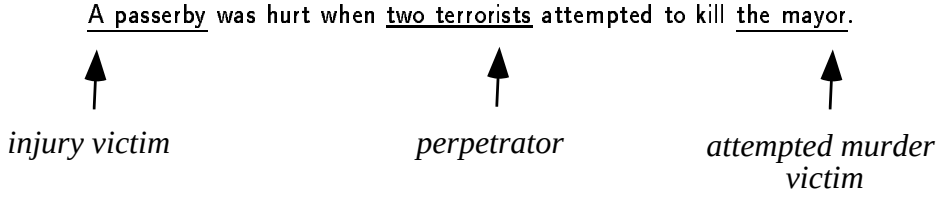
This dissertation addresses the knowledge-engineering bottleneck for domain-specific natural language processing systems. Automated knowledge acquisition is essential to the long-term success of knowledge-based natural language processing. We claim that dictionaries for some natural language processing tasks can be acquired with minimal effort, given a training corpus. In particular, we have developed a system called AutoSlog that uses a training corpus to build domain-specific dictionaries for a natural language processing task called "information extraction".

Information extraction systems are designed to extract specific types of information from text. For example, in the terrorism domain, an information extraction system might extract references to all perpetrators and victims of terrorist attacks. To illustrate, consider the following sentence:

A passerby was hurt when two terrorists attempted to kill the mayor.

An information extraction system for terrorism should extract "a passerby" as a victim, "two terrorists" as perpetrators, and "the mayor" as the victim of an attempted murder.

AutoSlog creates dictionaries of case frames that are used to extract information from text. The input to AutoSlog is a set of noun phrases and the output is a set of case frames through which these noun phrases can be extracted. AutoSlog uses a training corpus to drive the dictionary construction process, but the training corpus does not contain examples of case frames. Instead, the corpus contains examples of noun phrases that need to be extracted. For example, a training corpus for the terrorism domain would consist of texts in which all perpetrators and victims of terrorist attacks have been marked and labeled by a person. In the sentence above, "a passerby" would be marked as an injury victim, "two terrorists" would be marked as perpetrators, and "the mayor" would be labeled as the victim of an attempted murder, e.g.,



Given the annotated sentence as input, AutoSlog uses heuristics to generate case frames that can be used to extract the noun phrases. The case frames represent patterns that are general in nature and can be used to extract a general class of information, such as murder victims. Consequently, the case frames will be useful for extracting similar information from future texts. For example, given the annotated sentence above with the noun phrase "a passerby" marked as

an injury victim, AutoSlog generates a case frame to recognize the pattern "X was hurt". In the future, whenever the pattern "X was hurt" appears in a text, the case frame will extract X as an injury victim. For the previous sentence, AutoSlog generates three case frames that represent the following patterns:

```
If a text contains the expression 'X was hurt'
then extract X as a victim who was injured.

If a text contains the expression 'X attempted to kill'
then extract X as a perpetrator.

If a text contains the expression 'attempted to kill Y'
then extract Y as the victim of an attempted murder.
```

In recent years, there has been a great deal of interest in corpus-based techniques. Large corpora are a rich source of knowledge for automated knowledge acquisition. This dissertation presents several techniques for intelligently exploiting annotated corpora. The general idea is that a domain expert can quickly scan a pile of texts and annotate them. The annotations provide a source of domain knowledge that can be used to automatically produce domain-specific knowledge bases. The goal of this dissertation is to minimize the burden on the domain expert and to make the annotation process as simple and straightforward as possible. For example, the annotations required for AutoSlog can be done quickly and easily by anyone familiar with the domain. An annotated corpus for AutoSlog can be created in a matter of hours by a person with only minimal training. Equally important, the domain expert does not need any background in natural language processing. In the past, manually encoded knowledge bases were usually constructed by experienced natural language processing researchers. In contrast, annotated corpora for AutoSlog can be created by people with no background in text processing.

In Chapter 3, we describe AutoSlog in detail and present results that we have achieved with AutoSlog for several domains. In the terrorism domain, we compared a dictionary produced by AutoSlog with a dictionary that was manually constructed by two natural language processing researchers. We estimated that the hand-crafted dictionary required approximately 1500 person-hours to build. In contrast, the AutoSlog dictionary achieved 98% of the performance of the hand-crafted dictionary but took only 5 person-hours to build. Chapter 3 also presents results for AutoSlog in two additional domains, a joint ventures domain and a microelectronics domain, and describes two experiments in which novices created their own dictionaries for information extraction using AutoSlog.

1.2 Relevancy Signatures and Relevancy Indices

The second goal of this dissertation is to demonstrate that natural language processing techniques can be used effectively in information retrieval applications. As more documents become available on-line, intelligent information retrieval will become increasingly important to navigate the information highway efficiently and effectively. Traditional approaches to information retrieval use keyword searches and statistical techniques to retrieve relevant documents [Callan *et al.*, 1992, Foltz and Dumais, 1992, Frakes and Baeza-Yates, 1992, Salton, 1971, Salton, 1989, Turtle and Croft, 1991]. However, these approaches have well-known limitations and natural language processing techniques hold promise for overcoming many of them.

The information retrieval problem that we will focus on is *text classification*. The problem is the following: given a set of texts, the texts must be separated into two piles, one pile of texts that are *relevant* to a specific domain and one pile of texts that are *irrelevant* to the domain. For example, in the domain of terrorism, texts that mention a terrorist incident should be classified as

relevant and texts that do not mention a terrorist incident should be classified as irrelevant. This is a two-class text categorization task.

We propose that information extraction techniques can be used to support text classification. This approach represents a compromise between keyword-based techniques and in-depth natural language processing. We have developed several algorithms that use information extraction to classify texts on the basis of linguistic phrases and contexts. Information extraction technology is powerful enough to make discriminations that are difficult to make with keyword-based techniques, yet it is more robust and practical than in-depth natural language processing.

Keywords can recognize some concepts that are useful for relevancy discriminations. However, concepts are often expressed with complex linguistic patterns that cannot be recognized by keywords alone. To illustrate, we will use examples from the MUC-4 corpus [MUC-4 Proceedings, 1992]. The MUC-4 corpus consists of 1500 texts that mention something having to do with terrorism. However, the text classification task for MUC-4 (which will be described in Section 2.2) specifies that a text should be classified as "relevant" only if it mentions a specific terrorist incident involving civilian victims or targets. Only 53% of the texts in the MUC-4 corpus satisfy this definition of relevance. The remaining texts often refer to terrorism in general or describe incidents where terrorists are fighting against other terrorists or military forces. The classification task for MUC-4 involves, among other things, distinguishing texts that mention terrorism against civilian targets from texts that mention military incidents.

Although people are often killed in terrorist incidents, the word "dead" is not a good keyword for terrorism because people die in many ways that have nothing to do with terrorism. For example, in the MUC-4 corpus, the word "dead" commonly appears in both terrorist event descriptions and military event descriptions. However, the expression "was found dead" indicates that a dead body was found and has an implicit connotation of foul play. This expression is likely to appear in texts describing criminal activity and, in many cases, terrorist activity. For example, the expression "was found dead" appears in terrorist event descriptions but *never* appears in military event descriptions in the MUC-4 corpus. Consequently, the word "dead" is not a good keyword for terrorism by itself but the expression "was found dead" is a strong indicator of terrorism in the MUC-4 corpus.

The first text classification algorithm proposed in this dissertation is called the *Relevancy Signatures Algorithm*. A *signature* represents linguistic expressions that include linguistic context immediately surrounding an individual word. For example, signatures can represent active and passive verb constructions such as "has been bombed", or they can represent a noun preceded by a specific verb form, such as "was found dead" or "planted bombs". *Relevancy signatures* are signatures that are strongly associated with a domain. Relevancy signatures can be acquired automatically by identifying signatures that are highly correlated with a domain in a training corpus. For example, in the MUC-4 corpus, the signature representing the expression "was found dead" is highly correlated with texts describing terrorist events. Once a signature attains the status of a *relevancy signature*, it can be used to classify new texts. Relevancy signatures perform well in domains that are characterized by strong key phrases.

However, relevancy discriminations sometimes depend on the types of objects involved in an event. For example, assassinations of government officials are often politically motivated and are frequently carried out by terrorists. This is in contrast to assassinations of rock stars, which are usually carried out by random individuals. Similarly, car bombings are often the result of terrorist activity but bombings of military installations are usually the result of military actions. These sorts of discriminations depend on the objects involved in the event and cannot be made on the basis of a fixed set of linguistic expressions.

The second text classification algorithm that we will describe is called the *Augmented Relevancy Signatures Algorithm*. *Augmented relevancy signatures* represent relevant linguistic expressions as well as local semantic context surrounding them. Relevancy signatures alone represent relevant linguistic expressions, but they do not capture the semantic information surrounding the expression. For example, a signature can represent the expression "was bombed" but cannot represent *what* was bombed. Consider the following two sentences:

(a) A car bomb exploded.
(b) The foreign debt crisis exploded.

The first sentence probably describes a terrorist incident but the second one does not. This is because the object of the explosion in sentence (a) was a car bomb (which is typically associated with terrorism) but the object of the explosion in the second sentence was an abstract object, the foreign debt crisis.

Augmented relevancy signatures include semantic information about role objects as well as linguistic phrases. For example, given the sentence "The mayor of Achi was assassinated", augmented relevancy signatures represent the key phrase "was assassinated" as a signature and the object of the assassination, "the mayor of Achi", as a *government-official victim*. The incident is classified as relevant if both the key phrase, "was assassinated", and the object, *government-official victim*, are highly associated with the domain. As with the signatures, relevant types of objects are acquired automatically using a training corpus. Augmented relevancy signatures can achieve better results than relevancy signatures alone for domains in which the context surrounding key phrases is important.

Augmented relevancy signatures represent local context surrounding key phrases, but they only capture semantic context associated with a single object. Sometimes, relevancy discriminations depend on multiple pieces of information that may be scattered throughout a sentence. For example, consider the following sentence:

The man took the money and fled.

This sentence probably describes a robbery. However, none of the individual words or phrases are highly associated with robberies. The words "took", "money", and "fled" do not necessarily indicate a robbery. The phrase "took money" occurs in many contexts that do not describe a robbery, for example "the man took the money as a gift", "the man took the money out of his wallet", or "the man took the money from the customer". In fact, the word "fled" may be the most revealing word since people often flee from the scene of a crime. But people certainly flee from many different types of crimes, not just robberies. And sometimes the person who flees is a victim, not a perpetrator. This example illustrates how multiple pieces of information can act together to paint a clear picture of an event even though the words and phrases individually do not.

This phenomenon motivated the third text classification algorithm called the *Case-Based Text Classification Algorithm*. This algorithm uses case-based reasoning techniques (e.g., [Ashley, 1990, Hammond, 1986, Kolodner and Simpson, 1989] to represent bigger chunks of context for text classification. Instead of using individual linguistic phrases and objects to classify texts, the case-based algorithm uses the context of an entire sentence. To classify a new text, the algorithm looks at each sentence and uses the notion of a *relevancy index* to retrieve similar sentences from the training corpus. If the retrieved sentences are strongly associated with relevant texts in the training corpus, then the new text is classified as relevant. By using multiple pieces of information that may be scattered throughout a sentence, the case-based algorithm can recognize relevant event descriptions that the previous algorithms cannot. The case-based algorithm is most appropriate for domains that are not characterized by strong key phrases and depend on rich contextual distinctions.

The Relevancy Signatures Algorithm, the Augmented Relevancy Signatures Algorithm, and the Case-Based Text Classification Algorithm are knowledge-based approaches to text classification because they depend on a domain-specific information extraction system. However, information extraction systems for new domains can be developed quickly using AutoSlog. As a result, the text classification algorithms can be easily ported across domains. In the next section, we show how AutoSlog can be hooked up with the text classification algorithms to create *portable* text classification systems.

1.3 The Big Picture

In Section 1.1, we briefly described a system called AutoSlog that automatically constructs domain-specific dictionaries for information extraction. AutoSlog greatly reduces the knowledge-engineering bottleneck for information extraction systems. In this section, we describe how the dictionary created by AutoSlog can also be used to support higher-level language tasks.

In particular, dictionaries created by AutoSlog can be combined with statistical algorithms to create knowledge-based text classifiers. A picture of the general scheme appears in Figure 1.1. The input is an annotated training corpus of texts. The information that needs to be extracted from these texts has been marked by a person. The training corpus is then given to AutoSlog which builds a dictionary of case frames that can extract the desired types of information. These case frames constitute a dictionary for the domain.

Each text is then given to a conceptual sentence analyzer, CIRCUS. CIRCUS uses the dictionary produced by AutoSlog to extract information from the texts. Statistical techniques identify which types of information are highly correlated with the relevant texts in the training corpus. That is, the statistics reveal which kinds of information are more commonly found in relevant texts than in irrelevant texts. Presumably, these types of information are good indicators for the domain and can be used to recognize relevant texts in a new corpus.

All three text classification algorithms mentioned in the previous section can be hooked up with AutoSlog in this fashion. Each algorithm compiles statistics for different kinds of linguistic context. The relevancy signatures algorithm identifies which *signatures* produced by CIRCUS are highly correlated with relevant texts. The augmented relevancy signatures algorithm identifies which *signatures and extracted objects* are highly correlated with relevant texts. And the case-based algorithm identifies which *cases* (i.e., sentence contexts) are highly correlated with relevant texts.

It is important to keep in mind that the signatures and case base are all generated automatically as a side effect of CIRCUS' sentence processing. The statistical techniques are domain-independent, so the only domain-specific information used by the system is the dictionary of case frames.[1] The dictionary of case frames, however, can be constructed automatically by AutoSlog using a training corpus. As a result, the text classification system benefits from knowledge-based natural language processing but is portable across domains.

1.4 Research Contributions

The research contributions of this work fall into two categories: dictionary construction for information extraction, and the application of natural language processing to text classification.

Claim #1: Dictionaries for information extraction can be constructed automatically.

Information extraction systems typically rely on a dictionary of case frames or patterns to extract relevant information from text [Hobbs *et al.*, 1992, Jacobs *et al.*, 1991, Lehnert *et al.*, 1992a]. This dictionary is usually the primary knowledge-engineering bottleneck for information extraction systems. In the past, these dictionaries have been constructed manually by experienced natural language processing researchers.

Using AutoSlog, dictionaries for information extraction can be acquired automatically with only minimal effort. AutoSlog is a major contribution toward making information extraction systems portable across domains. In the terrorism domain, a dictionary produced by AutoSlog achieved performance comparable to that of a hand-crafted dictionary. We have also applied

[1] Well, almost. The augmented relevancy signatures algorithm and the case-based algorithm also rely on a semantic feature dictionary. We address the problem of how to acquire semantic features in Section 4.6.3.2.

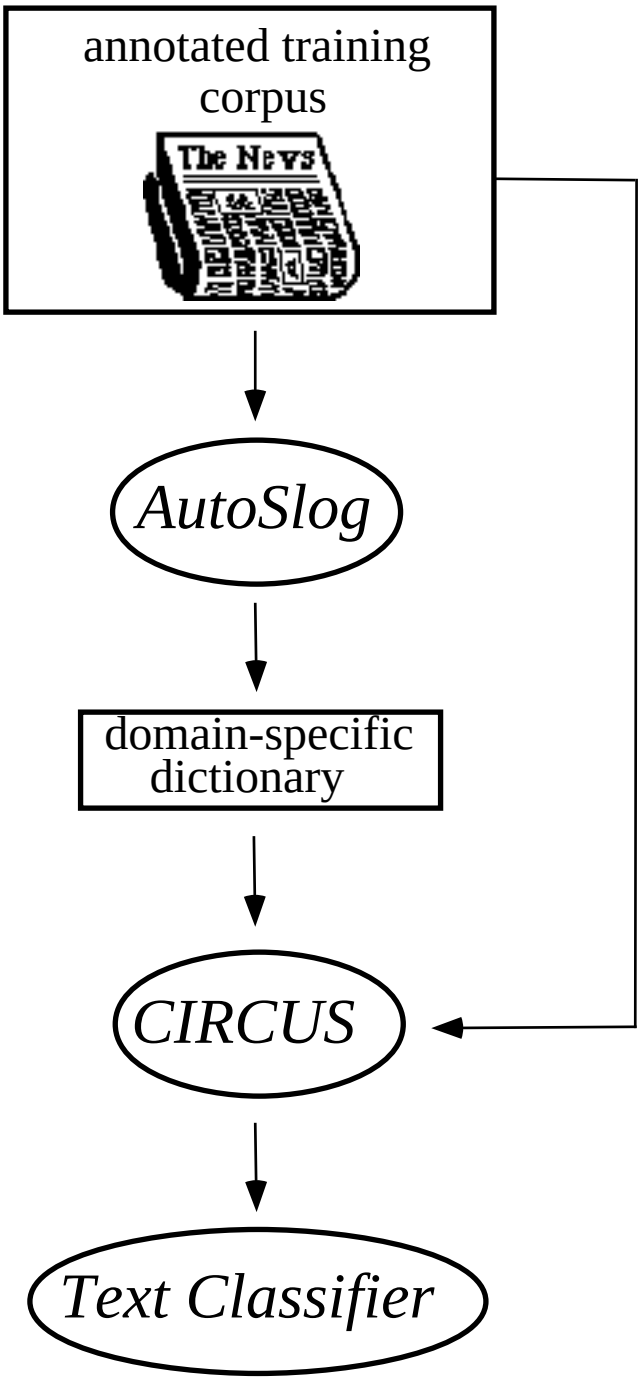


Figure 1.1: Flowchart for portable text classification systems

AutoSlog to two other domains, joint ventures and microelectronics, and achieved good results in these domains as well. We claim that the heuristics underlying AutoSlog are applicable to a broad range of domains.

Claim #2: Information extraction can support high precision text classification.

Traditional approaches to information retrieval use word-based techniques that have well-known limitations, which we will outline in Section 4.1. Natural language processing techniques can overcome many of these limitations but are often hampered by practical difficulties. The state-of-the-art in natural language processing is such that in-depth analyses of unconstrained text are not yet feasible. Information extraction, however, is a practical and robust technology that has achieved good success for domain-specific text analysis [Lehnert and Sundheim, 1991, MUC-3 Proceedings, 1991, MUC-4 Proceedings, 1992, MUC-5 Proceedings, 1993]. We claim that information extraction techniques are powerful enough to overcome many of the limitations of word-based techniques and can achieve strong performance on text classification tasks. In particular, information extraction techniques can be used to build highly accurate text classification systems.

The third claim of this dissertation is:

Claim #3: Knowledge-based text classification systems can be portable across domains.

The text classification algorithms that we have developed are knowledge-based because they rely on an information extraction system that uses a domain-specific dictionary. However, the text classification algorithms are domain-independent and the domain-specific dictionary can be acquired automatically, given an appropriate training corpus. Therefore the complete text classification system is fully trainable and can be easily scaled up or ported to new domains. By automating the construction of a knowledge-based text classification system, we have greatly reduced the knowledge engineering bottleneck typically required for such systems while still benefiting from a knowledge-based approach.

1.5 Guide to this Dissertation

To help the reader navigate this dissertation, Table 1.1 briefly describes the contents of each chapter. The annotations for each chapter (in italics) indicate the importance of the chapter and how it relates to the others.

Table 1.1: Guide to this dissertation

CHAPTER 2: Introduction to information extraction (IE), the CIRCUS sentence analyzer, and the MUC-4 and MUC-5 tasks. *Important background for readers who are not familiar with CIRCUS or MUC-4 and MUC-5.*
CHAPTER 3: Description of the AutoSlog system that automatically constructs dictionaries for information extraction; experimental results in three domains; experimental results for dictionaries created by novice users. *For readers who are interested in the technical details and results of automated dictionary construction. The sections on experiments with novice users can be skipped without loss of continuity.*
CHAPTER 4: Description of three text classification algorithms based on information extraction; a procedure for automatically identifying good threshold values; experimental results in three domains.[2] *For readers who are interested in the technical details and results of IE-based text classification algorithms.*
CHAPTER 5: Discussion of general properties of domains and tasks that are appropriate for AutoSlog and IE-based text classification; explanation of how to port the systems across domains. *For readers who are interested in the general applicability of the approaches and how to bring up the systems in a new domain.*
CHAPTER 6: Discussion of related work in automated dictionary construction, machine learning, information retrieval, and case-based reasoning. *For readers who are interested in how this research relates to previous work.*
CHAPTER 7: Summary of research claims, supporting results, and directions for future research. *Summary of the claims and contributions of this dissertation.*

[2] Portions of this chapter also appear in [Riloff and Lehnert, 1994].

CHAPTER 2

INFORMATION EXTRACTION

Natural language processing (NLP) holds promise for dealing with many real-world applications, such as database access, question answering, and information retrieval. However, in-depth natural language processing is an expensive endeavor that can strain computational resources. Furthermore, the state-of-the-art in natural language processing is such that we do not yet have practical NLP systems that can generate in-depth analyses of unconstrained text.

As an alternative to full-blown natural language processing, some researchers in the NLP community have turned their attention to *information extraction*. Whereas in-depth natural language processing requires a complete analysis of text, information extraction is a more focused and well-defined task. The goal of an information extraction system is to extract specific types of information from text. For example, in the domain of terrorism, an information extraction system might extract the names of all perpetrators, victims, physical targets, and weapons that were involved in a terrorist attack. The main advantage of this task is that portions of a text that are not relevant to the domain can be effectively ignored. This simplifies the job of the NLP system considerably. Information extraction is less computationally expensive than full-blown natural language processing because many phrases and even entire sentences can be ignored if they are not relevant to the domain. And since the system is only concerned with the *domain-specific* portions of the text, some of the most difficult problems in NLP are simplified (e.g., part-of-speech tagging, ambiguity resolution, etc.). Information extraction is a more practical and robust technology than in-depth natural language understanding and has achieved success in the last few years [Lehnert and Sundheim, 1991, MUC-3 Proceedings, 1991, MUC-4 Proceedings, 1992, MUC-5 Proceedings, 1993].

In this chapter, we describe a conceptual sentence analyzer called CIRCUS that performs information extraction, and we describe the MUC-4 and MUC-5 information extraction tasks and corpora that were used to evaluate the work in this dissertation.

2.1 Selective Concept Extraction Using CIRCUS

Selective concept extraction is a natural language processing technique that supports information extraction. This technique is essentially a form of text-skimming that selectively processes text that is relevant to a domain. Selective concept extraction is implemented in a conceptual sentence analyzer called CIRCUS [Lehnert, 1991]. The backbone of CIRCUS is a domain-specific dictionary of *concept nodes*. Concept nodes are case frames that extract relevant information from a sentence.

A concept node is triggered by an individual word but is activated only in certain linguistic contexts. Each concept node has a set of enabling conditions that specify a linguistic context that must be present in order for the concept node to be activated. For example, in the domain of terrorism, the CIRCUS dictionary contains two concept nodes that are both triggered by the word "murdered". The first one, $murder-active$, is activated if the verb "murdered" appears in

an active construction, such as "the terrorists murdered the mayor". The second concept node, $murder-passive$, is activated if the verb "murdered" appears in a passive construction, such as "three peasants were murdered by guerrillas". A concept node may be triggered by several different words. For example, $murder-passive$ is also triggered by the word "killed" so it is also activated by phrases such as "three peasants were killed by guerrillas". If a sentence contains multiple trigger words, then CIRCUS may produce multiple concept nodes for the sentence. If a sentence contains no trigger words, then CIRCUS produces no output for that sentence. Instantiated concept nodes are the only output generated by CIRCUS.

A concept node definition specifies a set of slots that extract information from text. Each slot extracts a particular type of information and contains a syntactic expectation that predicts where the information will be found in a clause. For example, $murder-passive$ contains two slots: a *victim* slot and a *perpetrator* slot. Since $murder-passive$ is activated only in a passive construction, the concept node predicts that the victim is the subject of the verb "murdered" and the perpetrator is the object of the preposition "by". Alternatively, $murder-active$ predicts that the subject of the verb is a perpetrator and the direct object is a victim. Figures 2.1 and 2.2 show the concept node definitions for $murder-active$ and $murder-passive$.

Name:	**$MURDER-ACTIVE$**
Trigger Word:	murdered
Variable Slots:	((perpetrator (*SUBJECT* 1)) (victim (*DOBJ* 1)))
Slot Constraints:	((class ORGANIZATION *SUBJECT*) (class TERRORIST *SUBJECT*) (class HUMAN *SUBJECT*) (class PROPER-NAME *SUBJECT*)) ((class HUMAN *DOBJ*) (class PROPER-NAME *DOBJ*))
Constant Slots:	(type murder)
Enabling Conditions:	((active))

Figure 2.1: The concept node definition for $MURDER-ACTIVE$

Each slot also has a set of hard and soft constraints that specify semantic preferences for the types of fillers that can legitimately fill the slot. The hard constraints must be satisfied in order for the slot to be filled. For example, $murder-passive$ contains a hard constraint (*is-prep?*) which dictates that perpetrators should be extracted only from prepositional phrases containing the preposition "by". Other prepositional phrases will not be extracted as perpetrators.

The soft constraints act only as preferences for fillers. Therefore a slot may be filled even if a soft constraint is violated. For example, $murder-passive$ contains soft constraints for both victims and perpetrators: victims should be of the class HUMAN or PROPER-NAME and perpetrators should be of the class ORGANIZATION, TERRORIST, HUMAN, or PROPER-NAME. A word satisfies the soft constraints for a slot if the word contains one of the appropriate semantic feature classes in its dictionary definition. CIRCUS uses a dictionary that associates semantic features with lexical items, for example the word "guerrilla" has the semantic feature TERRORIST in the dictionary,

Name:	**$MURDER-PASSIVE$**
Trigger Word:	murdered
Variable Slots:	(victim (*SUBJECT* 1)))
	(perpetrator (*PREP-PHRASE* (is-prep? '(by))))
Slot Constraints:	((class HUMAN *SUBJECT*)
	(class PROPER-NAME *SUBJECT*))
	((class ORGANIZATION *PREP-PHRASE*)
	(class TERRORIST *PREP-PHRASE*)
	(class HUMAN *PREP-PHRASE*)
	(class PROPER-NAME *PREP-PHRASE*))
Constant Slots:	(type murder)
Enabling Conditions:	((passive))

Figure 2.2: The concept node definition for $MURDER-PASSIVE$

and the word "peasant" has the semantic feature HUMAN.[1] A slot is filled even if the extracted information is not of the appropriate semantic class, but a semantic feature violation will be flagged. Words that are not in the dictionary are assumed to be proper nouns. Figure 2.3 shows a sample sentence and the resulting instantiated concept node produced by CIRCUS.

Sentence: Three peasants were murdered by guerrillas.

$MURDER-PASSIVE$
victim = "three peasants"
perpetrator = "guerrillas"

Figure 2.3: An instantiated concept node

Since concept nodes are the only form of CIRCUS output, the dictionary of concept nodes is crucial for effective information extraction. The UMass/MUC-4 system [Lehnert *et al.*, 1992a] used two dictionaries: a part-of-speech dictionary containing 5436 lexical definitions, including semantic features for domain-specific words, and a dictionary of 389 concept node definitions for the terrorism domain. The concept node dictionary was manually constructed for MUC-4, which is described in the next section. However, in Chapter 3 we explain how these concept node definitions can be acquired automatically [Riloff, 1993a, Riloff and Lehnert, 1993].

The part-of-speech and semantic feature dictionary was also built by hand for MUC-4. Each of the 5436 words in the dictionary was assigned one or more semantic features from the hierarchy shown in Figure 2.4. This hierarchy was derived from a concept hierarchy given to the participants of MUC-4 by the conference organizers. For example, the types of human targets, physical targets, and weapons were specified in the MUC-4 guidelines. We added a few features to represent concepts that are related to the relevant event types. For example, the feature *money* relates to robberies, and the feature *property* is used to distinguish small objects (such as windows and equipment)

[1]This does not imply that terrorists are not human. Terrorists are a subclass of humans in the semantic feature hierarchy.

from larger objects (such as buildings and vehicles) that are legitimate bombing targets. The semantic feature hierarchy contains some atypical categorizations; for example, *aerial-bombs* are not a subclass of bombs (because the MUC-4 guidelines categorized them separately) and *terrorists* are not a subclass of *human-target* because they were not considered to be legitimate victims of terrorist crimes. We added *clergy* as a special subclass of civilians because the MUC-4 corpus contained a lot of texts about the murder of several jesuit priests.

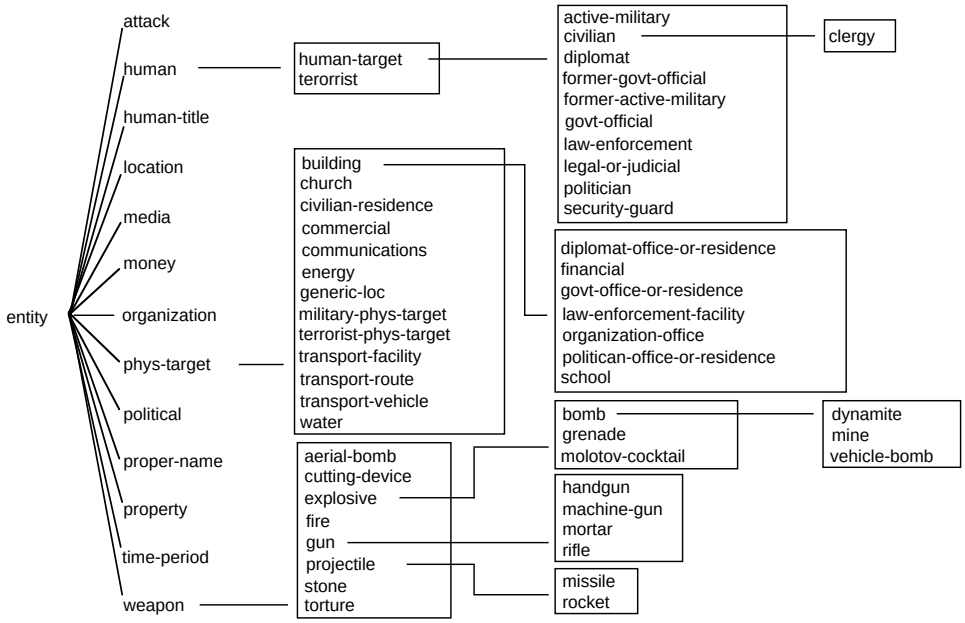


Figure 2.4: Semantic feature hierarchy for the terrorism domain

Some words in the dictionary were assigned multiple semantic features. For example, the definition of the word "army" contains both the *organization* and *active-military* features. Ambiguous words were also assigned multiple features, e.g., the definition of the word "Flores" contains three semantic features (*proper-name*, *human*, *location*) because it can be a person's name or a city.[2] The UMass/MUC-4 system used special routines to allow a head noun to inherit features from noun modifiers in the same noun phrase. Words that did not have an entry in the dictionary were assumed to be proper names.

It is important to note that the semantic feature hierarchy used by CIRCUS only represents concepts that are related to terrorism. The vast majority of the words in the dictionary were assigned the general *entity* feature. CIRCUS can get away with limited semantic coverage because it is a text-skimmer. Semantic features are accessed only by the concept nodes, inside enabling conditions or as soft constraints. For example, the murder concept nodes have soft constraints that prefer human victims, which allows CIRCUS to skip over metaphorical expressions such as "the bill was killed in Congress". Words that are not triggered or extracted by concept nodes are presumably unrelated to the domain anyway so the system does not need to know anything more about them.

[2]The MUC-4 dictionary did not distinguish between conjunctions of features (e.g., the army is both military and an organization) and disjunctions of features (e.g., Flores is either a human or a city).

Each of the words in the MUC-4 dictionary was also assigned one or more part-of-speech tags. The UMass/MUC-4 system used 12 parts-of-speech, one of which was a "special" tag that included several parts-of-speech such as relative pronouns and conjunctions that required more complicated processing. For MUC-5, CIRCUS employed a trainable part-of-speech tagger called OTB [Lehnert *et al.*, 1993a] that assigned parts-of-speech to words dynamically during sentence analysis. OTB used 18 part-of-speech tags, including the same "special" tag that covered additional parts-of-speech.

2.2 The MUC-4 Task and Corpus

Our interest in information extraction was motivated by the ARPA[3]-sponsored message understanding conferences. These conferences are competitive performance evaluations designed to assess the state-of-the-art in text analysis. The Fourth Message Understanding Conference (MUC-4) was held in June 1992. Seventeen research labs from both academia and industry participated in MUC-4. Each site had to develop a system to extract information about terrorism in Latin America from newswire articles. An extensive set of domain guidelines defined what constituted "terrorism". In general, a text was defined as relevant only if it mentioned a specific terrorist incident that occurred in one of seven Latin American countries. General descriptions of terrorist events (e.g., "there have been many bombings ..."), events that happened more than 2 months prior to the newswire date, and terrorist events involving military targets and personnel were not considered to be relevant. Eight general types of incidents were relevant: arsons, attacks, bombings, forced work stoppages, kidnappings, hijackings, murders, and robberies. Incidents of these types that were attempted (e.g., attempted murders) or threatened (e.g., death threats) were also relevant so there were actually 24 possible event types.

Figure 2.5 shows a relevant text from MUC-4. This text is relevant because it describes a terrorist attack on a civilian target (the U.S. embassy) in Miraflores, Peru.

DEV-MUC4-0042 (NCCOSC)
LIMA, 16 JAN 90 (TELEVISION PERUANA) – [TEXT] TEN TERRORISTS HURLED DYNAMITE STICKS AT U.S. EMBASSY FACILITIES IN THE MIRAFLORES DISTRICT, CAUSING SERIOUS DAMAGE BUT FORTUNATELY NO CASUALTIES. THE ATTACK TOOK PLACE AT 2100 ON 15 JANUARY [0100 GMT ON 16 JAN].

INSIDE THE FACILITY, WHICH WAS GUARDED BY 3 SECURITY OFFICERS, A GROUP OF EMBASSY OFFICIALS WERE HOLDING A WORK MEETING.

ACCORDING TO THE FIRST POLICE REPORTS, THE ATTACK WAS STAGED BY 10 TERRORISTS WHO USED 2 TOYOTA CARS WHICH WERE LATER ABANDONED. ONE OF THE VEHICLES WAS LEFT ON THE THIRD BLOCK OF JOSE PARDO AVENUE, WHILE THE OTHER WAS LEFT ON THE FIRST BLOCK OF BELLA VISTA STREET IN MIRAFLORES.

Figure 2.5: A relevant MUC-4 terrorism text

Figure 2.6 shows an irrelevant text from MUC-4. This text is irrelevant because it describes a military incident. Even though a terrorist group (the FMLN) was involved, this incident is not considered to be relevant because the terrorist group was fighting military forces.

[3]ARPA is the Advanced Research Projects Agency of the U.S. Government.

DEV-MUC4-0005 (NCCOSC)
CLANDESTINE, 8 JAN 90 (RADIO VENCEREMOS)– [TEXT] A WAR BULLETIN INDICATES THAT ON 6 JANUARY AT 1625, FMLN [FARABUNDO MARTI NATIONAL LIBERATION FRONT] TROOPS CLASHED WITH THE CAVALRY COMPANY IN FINCA SANTA ELENA, SANTA TECLA, NEAR SAN SALVADOR, KILLING THREE AND WOUNDING FOUR ENEMY TROOPS, INCLUDING THE PATROL LEADER WHO WAS AMONG THOSE KILLED. OUR TROOPS SEIZED AN M-14 RIFLE FROM THE ENEMY, 3,000 CARTRIDGES FOR A 7.62-MM RIFLE, FIVE KNAPSACKS, SIX GRENADES, AND FIELD EQUIPMENT.

Figure 2.6: An irrelevant MUC-4 terrorism text

Figure 2.7 shows another relevant text that is more typical of the MUC-4 texts. This text contains both relevant and irrelevant event descriptions. The second and third paragraphs describe two different bombing incidents that are both relevant because they involve terrorist perpetrators and civilian targets. The first, fourth, and fifth paragraphs, however, describe military incidents that are not relevant. A text is considered to be relevant if it mentions at least one relevant terrorist incident, even if it also contains irrelevant event descriptions.

For each relevant text, the MUC-4 systems had to extract relevant information and put the information into one or more "templates". A *template* is essentially a large case frame with a predefined set of slots, one slot for each type of information to be extracted from the text. For example, the MUC-4 templates had slots for the names of perpetrators, victims, physical targets, weapons, dates, locations, etc. A separate template had to be filled out for each different event. Each instantiated template was supposed to represent information pertaining to a single terrorist incident. If a text describes multiple relevant events then multiple templates should be generated. If a text describes no relevant terrorist events then only a dummy template containing no information should be generated. Figure 2.8 shows the instantiated bombing template for the relevant text in Figure 2.5.

The possible values for each slot fall into two different categories: string fills and set fills. The *string fill* slots are filled with strings that are extracted directly from the text (shown in quotes). For example, the **perp: individual id** slot is filled with the name of the perpetrator as it appears in the text; for text dev-muc4-0042 the perpetrators are "ten terrorists". Similarly, the human victims in text dev-muc4-0042 are "embassy officials" and "security officers". In general, the string fill slots have an infinite set of possible values.

In contrast, the *set fill* slots must be filled with a fixed set of predefined values. For example, the **incident: type** slot must be filled with one of the 8 event types: arson, attack, bombing, kidnapping, forced work stoppage, hijacking, murder, robbery. The **incident: stage of execution** slot must be filled with one of three possible values: accomplished, attempted, threatened. Together, these two slots effectively represent 24 types of events (e.g., murders, attempted murders, and death threats are different). Information often appears in multiple template slots as a string fill and set fills. For example, the instrument is represented as a string fill for the **instrument: id** slot ("dynamite sticks") and also as set fills for the **instrument: type** slot (dynamite). The set fill slots represent the general category of the string that appeared in the text. The distinction between string fill slots and set fill slots will become apparent when we discuss the automated dictionary construction system, AutoSlog, in Chapter 3.

ARPA provided the MUC-4 participants with a corpus of 1500 texts and associated answer keys to use for development purposes. The answer keys are instantiated templates that were

DEV-MUC4-0018 (NCCOSC)

SAN SALVADOR, 10 JAN 90 (AFP) – [TEXT] OFFICIAL SOURCES HAVE REPORTED THAT SEVERAL GUERRILLA ATTACKS AND HEAVY FIGHTING TOOK PLACE THE EVENING OF 9 JANUARY AND THIS MORNING THROUGHOUT THE COUNTRY, AND AS A RESULT, THREE SOLDIERS WERE KILLED AND THREE OTHERS INJURED.

ALLEGED GUERRILLA URBAN COMMANDOS LAUNCHED TWO HIGHPOWER BOMBS AGAINST A CAR DEALERSHIP IN DOWNTOWN SAN SALVADOR THIS MORNING. A POLICE REPORT SAID THAT THE ATTACK SET THE BUILDING ON FIRE, BUT DID NOT RESULT IN ANY CASUALTIES ALTHOUGH ECONOMIC LOSSES ARE HEAVY.

DURING THE EVENING OF 9 JANUARY, GUERRILLA URBAN COMMANDOS BOMBED TWO ELECTRICITY FACILITIES IN DIFFERENT PLACES IN SAN SALVADOR, WHICH CAUSED POWER OUTAGES IN SOME AREAS OF THE CAPITAL.

MEANWHILE, THE ARMED FORCES PRESS COMMITTEE (COPREFA) REPORTED TODAY THAT THREE ARMY SOLDIERS WERE KILLED RECENTLY IN CLASHES AGAINST MEMBERS OF THE FARABUNDO MARTI NATIONAL LIBERATION FRONT (FMLN) IN DIFFERENT PARTS OF THE CENTRAL AND EASTERN REGIONS OF THE COUNTRY.

THE WAR BULLETIN BY COPREFA STATED THAT THE CLASHES, IN WHICH THREE MEMBERS OF THE GENERAL JUAN RAMON BELLOSO BATTALION WERE INJURED, TOOK PLACE IN SAN JOSE GUAYABAL, IN THE CENTRAL CUSCATLAN DEPARTMENT, AND IN SANTA ELENA IN THE EASTERN USULUTAN DEPARTMENT.

Figure 2.7: Another relevant MUC-4 terrorism text

0.	**message: id**	dev-muc4-0042 (nccosc)
1.	**message: template**	1
2.	**incident: date**	15 Jan 90
3.	**incident: location**	Peru: Lima (city): Miraflores (neighborhood)
4.	**incident: type**	bombing
5.	**incident: stage of execution**	accomplished
6.	**incident: instrument id**	"dynamite sticks" / "dynamite"
7.	**incident: instrument type**	dynamite: "dynamite sticks" / "dynamite"
8.	**perp: incident category**	terrorist act
9.	**perp: individual id**	"ten terrorists" / "10 terrorists"
10.	**perp: organization id**	-
11.	**perp: organization confidence**	-
12.	**phys tgt: id**	"embassy facilities"
13.	**phys tgt: type**	diplomat office or residence: "embassy facilities"
14.	**phys tgt: number**	1 / plural: "embassy facilities"
15.	**phys tgt: foreign nation**	United States: "embassy facilities"
16.	**phys tgt: effect of incident**	some damage: "embassy facilities"
17.	**phys tgt: total number**	-
18.	**hum tgt: name**	-
19.	**hum tgt: description**	"embassy officials" "security officers"
20.	**hum tgt: type**	diplomat: "embassy officials" security guard: "security officers"
21.	**hum tgt: number**	plural: "embassy officials" 3: "security officers"
22.	**hum tgt: foreign nation**	United States: "embassy officials"
23.	**hum tgt: effect of incident**	no injury or death: "security officers" no injury or death: "embassy officials"
24.	**hum tgt: total number**	-

Figure 2.8: The MUC-4 terrorism template for text DEV-MUC4-0042

manually encoded by the participants of MUC-3[4] and MUC-4. Each answer key contains the correct information corresponding to a relevant terrorist incident reported in a text; that is, the information that should be extracted from the text. ARPA also supplied an additional 200 texts and associated answer keys as test sets for the final MUC-4 evaluation. This entire set of 1700 texts and corresponding answer keys served as the testbed for the experiments described in this dissertation.

The answer keys also served as a set of correct classifications for each text. If a text has instantiated key templates associated with it in the corpus, then it should be classified as a relevant text. If a text has no instantiated key templates associated with it (i.e., only a dummy template) then it should be classified as an irrelevant text. This is a binary classification problem: a text is either *relevant* to the terrorism domain or *irrelevant*. The texts were selected by keyword search from a database of newswire articles[5] because they contain words associated with terrorism. However, many of them do not mention any relevant terrorist incidents. Of the 1700 texts in the MUC-4 corpus, only 53% describe a relevant terrorist event.[6]

Because many of the texts in the corpus are irrelevant, the MUC-4 systems had to distinguish the relevant from the irrelevant texts. Although the MUC-4 task was information extraction, *information detection*[7] (i.e., text classification) was an implicit subtask. To be successful in MUC-4, the information extraction systems also had to be good at detection. The UMass/MUC-4 system did not use a separate text classification module. Instead, it extracted information from every text and relied on a discourse analysis module to discard irrelevant templates. This strategy was very effective[8], but it is expensive. A reliable text classification module could have filtered out irrelevant texts so we would not have needed to apply the complete NLP system to every text.[9] Furthermore, a text classification module could have improved accuracy by preventing irrelevant texts from slipping through to discourse analysis which often had trouble recognizing irrelevant event descriptions. Furthermore, the discourse analysis module was domain-specific and not portable across domains. The MUC-4 application illustrates how text classification can be useful not only for stand-alone applications, but also as a partner for other natural language processing tasks.

---

[4]MUC-3 was the Third Message Understanding Conference held in 1991 [MUC-3 Proceedings, 1991].

[5]The database was constructed by the Foreign Broadcast Information Service (FBIS) of the U.S. government [MUC-4 Proceedings, 1992].

[6]For our text classification experiments, we counted all of the texts that had "optional" templates as relevant texts.

[7]ARPA has recently initiated a competitive performance evaluation called TREC that focuses explicitly on the task of *information detection* [Harman, 1994, Harman, 1993].

[8]The MUC-4 systems were evaluated on the basis of two blind test sets, TST3 and TST4. The UMass/MUC-4 text filtering scores were 91% recall with 94% precision on TST3 and 91% recall with 82% precision on TST4 [MUC-4 Proceedings, 1992].

[9]The text classification algorithms described here still require CIRCUS to extract information from the texts, but the UMass/MUC-4 system contained additional components for discourse analysis that would not be needed.

2.3 The MUC-5 Tasks and Corpora

In 1993, ARPA sponsored the Fifth Message Understanding Conference (MUC-5). MUC-5 was also a competitive performance evaluation that focused on the information extraction task. ARPA gave the MUC-5 participants two new domains: joint ventures (a business domain) and microelectronics (a technical domain). Nineteen sites participated in MUC-5, including the NLP group at the University of Massachusetts. In the next two sections, we describe each of these domains.

2.3.1 The Joint Ventures Domain

The first domain is a business domain related to joint venture activities. For the joint ventures (JV) domain, a text is considered to be relevant if it describes a joint venture between two or more partners. A joint venture, or tie-up, is defined as: "a cooperative association between 2 or more parties to own and/or develop a project together". The names of at least two partners must be explicitly mentioned in the text and a business purpose or explicit identification as a tie-up or joint venture is necessary. A new corporate entity (i.e., a child company) may or may not result. Figure 2.9 shows a relevant JV text from the MUC-5 corpus.

0083: DECEMBER 18, 1990, TUESDAY Copyright (c) 1990 Jiji Press Ltd.
NIPPON FIRE AND MARINE INSURANCE CO. SAID TUESDAY IT WILL SET UP A NONLIFE INSURANCE FIRM IN JAKARTA WITH PT BANK BALI OF INDONESIA. NIPPON FIRE AND MARINE INSURANCE HAS ALREADY MADE A 49 PCT CAPITAL PARTICIPATION IN AMB, A NONLIFE INSURANCE FIRM UNDER THE CONTROL OF THE INDONESIAN BANK. AMB WILL INCREASE ITS CAPITAL ON THURSDAY FROM THE PRESENT 500 MILLION INDONESIAN RUPIAHS TO 15 BILLION RUPIAHS, OF WHICH 49 PCT WILL BE PROVIDED BY NIPPON FIRE AND MARINE. UPON OBTAINING INDONESIAN GOVERNMENT APPROVAL OF THE CAPITAL PARTICIPATION, AMB WILL START BUSINESS AS A JOINT NONLIFE INSURANCE COMPANY NAMED BALI NIPPON INSURANCE. NIPPON FIRE AND MARINE WILL BE THE FIFTH JAPANESE NONLIFE INSURANCE COMPANY TO SET UP SUCH A JOINT VENTURE IN THE ISLAND REPUBLIC.

Figure 2.9: A relevant MUC-5 joint ventures text

Irrelevant texts in the MUC-5 corpus often contain the phrase "joint venture" but do not mention a *specific* tie-up between partners. Figure 2.10 shows an example of an irrelevant text. This text contains the phrase "Joint Venture Committee" and reports on joint investments in general but does not describe a specific joint venture. Texts that mention a specific joint venture but name only one of the partners are also considered to be irrelevant. Other common types of irrelevant texts describe business activities that are similar in nature but do not satisfy the definition of a tie-up, such as mergers, acquisitions, activities of venture capitalists, etc.

The information extraction task for the joint ventures domain required the ability to extract several types of information, including the following:[10]

[10]The MUC-5 task also required the ability to extract additional information, such as locations, dates, etc. See [MUC-5 Proceedings, 1993] for a full description of the MUC-5 task.

2512: March 11, 1981, Wednesday
Copyright (c) 1981 Jiji Press

Japanese and Philippine business leaders at their meeting in this western Japanese city Tuesday and Wednesday agreed to consider establishing a Joint Investment Fund.

A joint statement of the Eighth Annual Conference of the Japan-Philippine Economic Co-operation Committee said a small business subcommittee will be set up in each other's Joint Venture Committee to promote Japanese small businesses' investment in the Philippines.

The Joint Fund scheme will be studied by the subcommittees, it said.

The two sides also agreed to hold the Ninth Conference in suburban Manila in March next year.

Speaking to newsmen after the committee meeting, Noboru Gotoh, Chairman of the Japanese National Committee for the Joint Body and President of Tokyu Corp., said Japan and the members of the Association of Southeast Asian Nations (ASEAN) will reach final agreement to set up ASEAN- Japan Development Corp. (AJDC) at a meeting in Singapore next Friday.

AJDC, which will be based in Singapore, is designed to promote Japanese firms' advancement into the ASEAN countries by financing, affording guarantees and extending consulting services for them.

It will be capitalized at two billion yen (about 10 million dollars), which will be equally put up by Japan and the ASEAN nations – Thailand, Malaysia, Indonesia, Singapore and The Philippines.

Gotoh also said AJDC will be formally inaugurated next June and start activities late this year.

Figure 2.10: An irrelevant MUC-5 joint ventures text

Entity:	name of a partner or the child company. (a company, government, or person)
Facility Name:	physical facility associated with an entity.
Ownership Percent:	percent of ownership by an entity.
Total Capitalization:	total (cash) capitalization of the tie-up.
Person Name:	name of a person associated with an entity.
Product/Service:	product or service that will result from the tie-up.
Revenue Rate:	expected revenue rate.
Revenue Total:	total expected revenue for a period of time.

As in the terrorism domain, the MUC-5 systems had to put the extracted information into one or more templates. However, the JV templates were object-oriented and included explicit links between objects. The details of the template structures are not important here (see [MUC-5 Proceedings, 1993]); a simplified version of a JV template is shown in Figure 2.11, corresponding to the text in Figure 2.9. The template shows a tie-up relationship between the Nippon Fire and Marine Insurance Co. and the PT Bank Bali. As a result of the tie-up, a child company called AMB was formed. AMB is an insurance company which is classified as a financial service. AMB was initially capitalized at 50 million rupiahs (49% owned by Nippon Fire and Marine Insurance) and then increased its capital to 15 billion rupiahs (49% from Nippon Fire and Marine Insurance). The template notation is confusing, but it is clear where the numbers come from in the text.

The MUC-5 development corpus for the joint ventures domain contains 999 texts and corresponding answer keys for each text. The corpus also contains 3 tests sets of 100 texts each (Tips1, Tips2, and Tips3) that were used for various testing phases during MUC-5.

2.3.2 The Microelectronics Domain

The second domain tested in MUC-5 was microelectronics. In general, microelectronics is a broad area that involves many different aspects of microchip fabrication. However, only certain stages and processes are relevant to the MUC-5 task. In particular, only texts that contain information about layering, lithography, or etching in wafer fabrication, or texts that mention packaging are considered to be relevant. These four microelectronics activities (layering, lithography, etching, and packaging) are referred to as *processes*. In addition, a text is relevant only if a process is linked to a specific company or research group. Figure 2.12 shows a relevant microelectronics text from the MUC-5 corpus.

In contrast, Figure 2.13 shows an irrelevant MUC-5 text. This text contains a keyword, "etching", that is associated with one of the relevant microelectronics processes. However, this text describes etching in the context of vehicle glass, so the text is not about microelectronics at all.

As before, the MUC-5 microelectronics systems had to put the extracted information into one or more templates. The ME templates were also object-oriented with many links between objects. A simplified version of an ME template is shown in Figure 2.14 (see [MUC-5 Proceedings, 1993] for details). This template corresponds to the text in Figure 2.12. The template shows a microelectronics capability involving a company called "General Semiconductor Industries Inc.". General Semiconductor is shown as a purchaser or user of a packaging type called DIP (dual in-line package). The packaging design has P_L_counts (pin or lead counts) of 8 and 16 pins. Finally, the objects being packaged are "diode arrays" that are classified as ACTIVE_DISCRETE_DEVICES. Note that "diode arrays" are listed only in a comment slot and do not actually have to be extracted by the system. The system merely has to classify the packaging object into one of the predefined categories, in this case, ACTIVE_DISCRETE_DEVICES.

The MUC-5 information extraction systems were required to extract many different kinds of information about microelectronics activities. However, there is one major difference between the previous information extraction tasks for terrorism and JV, and the information extraction

```
Template Summary:  0083;  181290;  "Jiji Press Ltd"
 Tie-up:  EXISTING
   Entities:
     COMPANY:  NIPPON FIRE AND MARINE INSURANCE CO;
       Aliases:  "NIPPON FIRE AND MARINE INSURANCE" / "NIPPON FIRE AND MARINE"
       Nationality: Japan  (COUNTRY)
       Ownership:
         Ownership-%: 49
         Total-capitalization: 500000000 IDR
       Ownership:
         Ownership-%: 49
         Total-capitalization: 15000000000 IDR
     COMPANY:  PT BANK BALI
       Location: Indonesia (COUNTRY)
       Nationality: Indonesia (COUNTRY)
   Joint Venture Co:
     COMPANY:  AMB;
       Aliases:  "BALI NIPPON INSURANCE"
       Location: Jakarta (CITY 1) Jakarta Raya (PROVINCE) Indonesia (COUNTRY)
   Activity:
     Industry:
       Finance: "A JOINT NON LIFE [INSURANCE COMPANY] NAMED BALI NIPPON INSURANCE" /
                "A [NON LIFE INSURANCE FIRM] UNDER THE CONTROL OF THE INDONESIAN BAN" /
                "A NONLIFE [INSURANCE FIRM]"
     Site:
       COMPANY:  AMB;
         Aliases:  "BALI NIPPON INSURANCE"
         Location: Jakarta (CITY 1) Jakarta Raya (PROVINCE) Indonesia (COUNTRY)
```

Figure 2.11: The MUC-5 joint ventures template for text 0083

General Semiconductor Industries, Inc. (GSI), announced production of new 8-pin and 16-pin dual-in-line package diode arrays. The DA series uses TransZorb (R) TVS technology to provide transient voltage suppression at low clamping voltages for sensitive data lines. These devices are applicable in the protection of board-level data communications, power and/or data bus lines. Both the 8-pin and 16-pin DIP diode arrays feature 500 watt peak pulse power per line for board level transients, standard DIP packaging and common ground configuration.

Figure 2.12: A relevant MUC-5 microelectronics text

Glass Medic, maker and marketer of service and repair systems for damaged vehicle glass, will open an operations center in Maulden, UK. Operational: 8/90. The facility will centralize inventory storage and sales and service for Glass Medic's UK business as well as new opportunities in Scandanavia and W Europe. Glass Medic has a theft-prevention system, SecurEtch, for etching vehicular glass.

Figure 2.13: A irrelevant MUC-5 microelectronics text

```
Template Summary:  2547486;  260290;  "News Release"
 MICROELECTRONICS CAPABILITY:
    PROCESS:
       PACKAGING:
          Type: DIP;
          P_L_Count: 8, 16;
       DEVICE:
          Function: ACTIVE_DISCRETE_DEVICES;
          Comment: ''diode arrays'';
    PURCHASER_OR_USER:
       ENTITY: Company:  General Semiconductor Industries Inc.;
```

Figure 2.14: The MUC-5 joint ventures template for text 0083

task for microelectronics. In terrorism and JV, most of the important information was stored in the templates as string fills (e.g., "armed guerrillas"). In microelectronics, most of the important information was stored as set fills (e.g., ACTIVE_DISCRETE_DEVICES). The MUC-5 systems still had to identify portions of the text that contained relevant information (e.g., "diode arrays") but only the general category of the information (ACTIVE_DISCRETE_DEVICES) was recorded in the template.

The work described in this dissertation is concerned with 12 types of information (corresponding to template slots) that need to be extracted from relevant microelectronics texts. Of the 12 slots, only two of them (entity name and equipment name) contain string fills; the remaining slots contain set fills.

Bonding Type:	tape automated bonding, laser/wire bonding, flip chip.
Device Size:	the capacity or complexity of the device.
Device Speed:	clock frequency, clock speed, or access time.
Device Function:	active discrete, microprocessor, ASIC, gate array, memory.
Entity Name:	a company, government, or person that plays the role of *developer*, *manufacturer*, *distributor*, *purchaser*, or *user*.
Equipment Name:	name or model number of equipment.
Equipment Type:	oxidation system, deposition system, epitaxial system, lithography system, etching system, tape automated bonder, or modular equipment.
Film Type:	insulators, semiconductors, metal.
Granularity Size:	line width, resolution, gate size, or feature size.
Material Type:	ceramic, plastic, epoxy, glass, ceramic glass.
Pin Count:	the number of pins or leads in the packaging design.
Process Type:	layering, lithography, etching, or packaging.

The microelectronics domain is fundamentally different from the terrorism and joint ventures domains because it is a technical domain. We will explain some of the differences between technical domains and event-based domains in Chapter 5 and discuss which types of domains are most well-suited for our algorithms.

2.4 Summary

In this section, we described the following:

- A conceptual sentence analyzer called CIRCUS that performs information extraction.
- The MUC-4 information extraction task for the terrorism domain and the MUC-4 corpus.
- The MUC-5 information extraction tasks for the joint ventures and microelectronics domains and the MUC-5 corpora.

CHAPTER 3

AUTOMATED DICTIONARY CONSTRUCTION FOR INFORMATION EXTRACTION

3.1 Motivation

Knowledge-based natural language processing systems have demonstrated good performance for information extraction tasks in limited domains [Lehnert and Sundheim, 1991, MUC-3 Proceedings, 1991, MUC-4 Proceedings, 1992, MUC-5 Proceedings, 1993]. But enthusiasm for their success is often tempered by real-world concerns about portability and scalability. Knowledge-based NLP systems depend on a domain-specific dictionary that must be carefully constructed for each domain. Building this dictionary is typically a time-consuming and tedious process that requires many person-hours of effort by highly-skilled people who have extensive experience with the system. Dictionary construction is a major knowledge engineering bottleneck that needs to be addressed in order for information extraction systems to be portable and practical for real-world applications.

We have developed a program called AutoSlog that automatically constructs domain-specific dictionaries for information extraction. Given a training corpus, AutoSlog proposes a set of dictionary entries that are capable of extracting specific types of information from the training texts. If the training corpus is representative of the domain, the dictionary created by AutoSlog will achieve strong performance for information extraction from novel texts. Given a training set from the MUC-4 corpus, a dictionary created by AutoSlog for the terrorism domain achieved 98% of the performance of a hand-crafted dictionary on two blind test sets. We estimate that the hand-crafted dictionary required approximately 1500 person-hours to build. In contrast, the AutoSlog dictionary was constructed in only 5 person-hours given a training corpus. Furthermore, constructing a dictionary by hand requires a great deal of training and experience whereas a dictionary can be constructed using AutoSlog with only minimal training.

In Section 3.2, we describe some key ideas that led to the development of AutoSlog. Section 3.3 explains how AutoSlog generates concept node definitions for a domain using a training corpus. In Section 3.4, we present examples of both good and bad concept node definitions produced by AutoSlog for the terrorism domain. Section 3.5 presents empirical results for AutoSlog in three different domains: terrorism, joint ventures, and microelectronics. Finally, Section 3.6 describes two experiments with novices which demonstrate that people with no background in text processing can successfully use AutoSlog.

3.2 Behind the Design of AutoSlog

Two observations were central to the design of AutoSlog. The first observation is that news reports follow certain stylistic conventions. In particular, the most important facts about a news event are usually reported first. Details and secondary information are described later. It follows that the first reference to an important object related to an event (e.g., a victim or perpetrator) usually appears in a sentence that mentions the event. For example, a story about the kidnapping of a diplomat probably mentions that the diplomat was kidnapped before it provides background information about the diplomat's family, etc. To put it another way, the first reference to the diplomat probably mentions the kidnapping. This observation is key to the design of AutoSlog. AutoSlog operates under the assumption that the *first* reference to an important object in an event is most likely where the relationship between that object and the event is made explicit.[1]

Once AutoSlog has identified the first sentence that mentions an object involved in an event, it determines which words or phrases should activate a concept node that can be used to extract the noun phrase that refers to the object. The second key observation behind AutoSlog is that the immediate linguistic context surrounding the noun phrase usually contains the words or phrases that describe the role of the object in the event. For example, consider the sentence "A U.S. diplomat was kidnapped by FMLN guerrillas today". This sentence contains two important pieces of information about the kidnapping: the victim ("U.S. diplomat") and the perpetrator ("FMLN guerrillas"). In both cases, the word "kidnapped" is the key word that relates them to the kidnapping event. In its passive form, we expect the subject of the verb "kidnapped" to refer to a victim and we expect the prepositional phrase beginning with "by" to contain a noun phrase that refers to a perpetrator. The word "kidnapped" specifies the roles of these people in the kidnapping and is therefore the most appropriate word to trigger a concept node.

AutoSlog relies on a small set of heuristics to determine which words and phrases are likely to activate useful concept nodes. In the next section, we will describe these heuristics and explain how AutoSlog generates complete concept node definitions.

3.3 The Algorithm

AutoSlog's job is to automatically build a dictionary of concept node definitions that can be used to extract certain types of information from text. For example, in the domain of terrorism, AutoSlog builds a dictionary of concept nodes that can be used to extract the names of perpetrators, victims, physical targets, and weapons. AutoSlog's strategy is to look at samples of information that needs to be extracted and, for each one, propose a concept node that can be used to extract the information. A set of heuristics is used to generate the concept nodes. The resulting concept node definitions are general in nature so they are applicable to new texts as well.

For the experiments described in Section 3.5, the input to AutoSlog is a set of texts and answer keys that contain the desired information that needs to be extracted from the texts. It is important to emphasize that these answer keys are not a requirement for AutoSlog. The answer keys contain a lot of additional information that AutoSlog does not need or use. In theory, AutoSlog requires only an annotated corpus in which the information that needs to be extracted from each text has been marked and labeled with semantic tags. We will return to this point in Section 5.4.3. However, for all the experiments described in this chapter, we used the answer keys as training data.

---

[1]The current implementation of AutoSlog is based on this assumption, but we emphasize that this assumption is **not** necessary if an annotated corpus is available. Section 5.4.3 describes how an annotated corpus would contain markings that make this assumption unnecessary.

Given a set of training texts and marked noun phrases as input[2], AutoSlog proposes a set of concept node definitions that can be used to extract the noun phrases from the corresponding texts. Since the concept node definitions are general in nature, we expect that many of them will be useful for extracting information from novel texts as well. The algorithm for constructing concept node definitions is as follows. For each targeted noun phrase (represented as a text string), AutoSlog finds the first sentence in the text that contains the string. This step is based on the observation noted earlier that the first reference to an object involved in an event is likely to be the place where the object is explicitly related to the event.[3]

The sentence is then handed over to CIRCUS which identifies the main syntactic constituents of the sentence. For example, CIRCUS breaks the sentence into clauses and identifies the subject, verb, direct object, and prepositional phrases of each clause. Using this information, AutoSlog identifies the first clause in the sentence that contains the string. Given the appropriate clause, a set of heuristics is applied to suggest a *conceptual anchor point* for a concept node definition. If none of the heuristics is satisfied then AutoSlog searches for the next sentence in the text that contains the string and the process is repeated. Figure 3.1 shows the general architecture of AutoSlog.

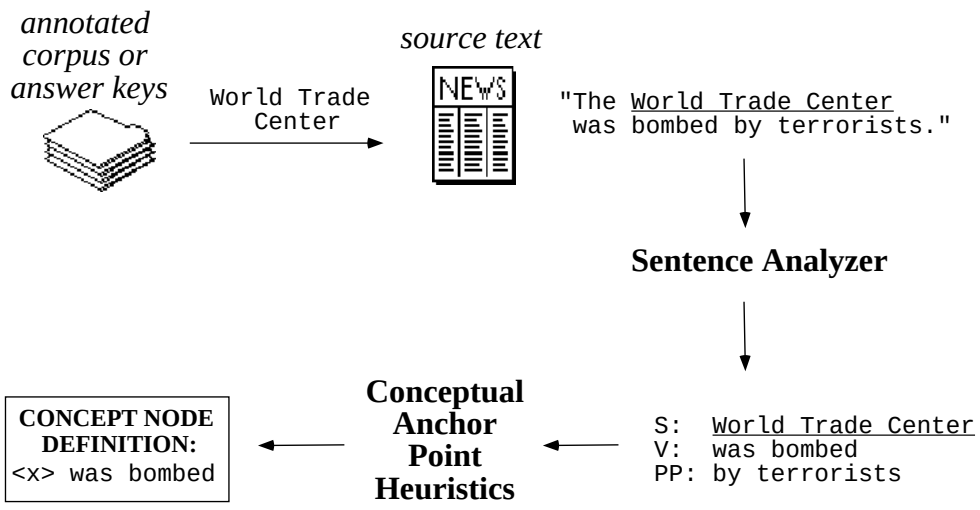


Figure 3.1: AutoSlog flowchart

The *conceptual anchor point heuristics* are the heart of AutoSlog. A conceptual anchor point is a word that should activate a concept; this is the trigger word of a concept node definition. The heuristics are divided into three categories depending upon where the targeted string is found in the clause. The string must be a noun phrase, so there are three possibilities: it could be the

[2]Either in the form of an annotated corpus or a set of texts and answer keys.

[3]This step is not necessary if AutoSlog uses an annotated corpus because AutoSlog can automatically identify which sentences contain the marked noun phrases. However, for the experiments in this dissertation we used the MUC-4 and MUC-5 answer keys as input and they did not identify which sentences the noun phrases came from in the source texts.

subject, the direct object, or in a prepositional phrase.[4] AutoSlog uses a different set of heuristics for each of these three cases.

If the targeted noun phrase is the subject of the clause, then AutoSlog assumes that the verb should be the conceptual anchor point. That is, AutoSlog assumes that the verb is the word that relates the object to an event. For example, given the noun phrase "the mayor" and the clause "the mayor was kidnapped", AutoSlog assumes that the verb "kidnapped" is the word that relates the mayor to the kidnapping event. If the targeted noun phrase is the direct object of the clause, then AutoSlog also assumes that the verb should be the conceptual anchor point. For example, given the noun phrase "the U.S. embassy" and the clause "terrorist bombed the U.S. embassy", AutoSlog assumes that the verb "bombed" is the word that relates the U.S. embassy to the bombing event.

If the targeted noun phrase is in a prepositional phrase, then AutoSlog uses a prepositional phrase attachment algorithm to identify the conceptual anchor point. In this case, AutoSlog assumes that the attachment point for the prepositional phrase is the word that relates the noun phrase to the event. For example, given the noun phrase "armed men" and the clause "five people died during a robbery yesterday in Bogota by armed men", a prepositional phrase algorithm should attach the "armed men" to the word "robbery".[5]

The heuristics, however, do more than just identify a trigger word. The heuristics also look for linguistic patterns. Each heuristic looks for a different linguistic pattern surrounding the trigger word. For example, verbs come in many forms; a verb can be a passive verb, an active verb, an auxiliary verb, an infinitive verb, or a gerund. Each of these verb forms represents a different pattern. If a heuristic successfully finds its pattern in the clause then it generates two things: (1) a conceptual anchor point and (2) a set of enabling conditions to recognize the pattern. For example, suppose AutoSlog is working on the clause "the diplomat was kidnapped" with "the diplomat" as the targeted noun phrase. "The diplomat" is the subject of the clause and is followed by a passive form of the verb "kidnapped". One of the heuristics recognizes the pattern **<subject> passive-verb**. Given this example, the heuristic fires and returns the word "kidnapped" as the conceptual anchor point along with enabling conditions that require a passive verb form. The result is a concept node definition that acts like the following rule: if the expression "X was/were/have been kidnapped" appears in a text, then extract X as the victim of a kidnapping.

The original version of AutoSlog used 13 heuristics, each designed to recognize a different linguistic pattern. These patterns are shown in Figure 3.2, along with examples that illustrate how they might appear in a text. The bracketed item shows the syntactic constituent where the targeted noun phrase was found. This syntactic constituent is used for the slot expectation. In the examples on the right, the bracketed item is a slot name that might be associated with the filler (e.g., the subject is a victim). The underlined word is the conceptual anchor point that is used as the trigger word.

---

[4]Theoretically, there are other possibilities (such as indirect objects) but these are the only syntactic constituents for noun phrases that are recognized by CIRCUS.

[5]The prepositional phrase attachment algorithm in AutoSlog is separate from CIRCUS and is very simple. If the preposition is "of", "against", or "on", then the algorithm attaches the prepositional phrase to the most recent constituent; otherwise, the algorithm attaches the prepositional phrase to the most recent verb or noun phrase but skips over intervening prepositional phrases. This algorithm makes a lot of mistakes and was intended only as a simple attempt to handle pp-attachment. A (very) slightly more sophisticated pp-attachment algorithm was used for the joint ventures and microelectronics domains (see Section 3.5.2.2).

[6]In principle, passive verbs should not have objects. However, we included this pattern because CIRCUS occasionally confuses active and passive constructions.

Linguistic Pattern	Example
<subject> passive-verb	<victim> was murdered
<subject> active-verb	<perpetrator> bombed
<subject> verb infinitive	<perpetrator> attempted to kill
<subject> auxiliary noun	<victim> was victim
passive-verb <direct-object>[6]	killed <victim>
active-verb <direct-object>	bombed <target>
infinitive <direct-object>	to kill <victim>
verb infinitive <direct-object>	threatened to attack <target>
gerund <direct-object>	killing <victim>
noun auxiliary <direct-object>	fatality was <victim>
noun preposition <noun-phrase>	bomb against <target>
active-verb preposition <noun-phrase>	killed with <instrument>
passive-verb preposition <noun-phrase>	was aimed at <target>

Figure 3.2: AutoSlog heuristics and examples for the terrorism domain

To illustrate the process, consider the case where the targeted noun phrase is the subject of a clause. In this case, the first four **<subject>** heuristics in Figure 3.2 are activated. AutoSlog assumes that the verb is the word that describes the role of the object in the event, but the verb might be in a passive construction, an active construction, or it could be an auxiliary verb.[7] If the verb is in a passive construction, then the first heuristic kicks in and proposes a concept node to recognize the pattern **<subject> passive-verb**, such as "<x> was murdered". If the verb is in an active construction (and is not an auxiliary verb), then two possible heuristics might apply. If the verb is immediately followed by another infinitive verb, e.g. "attempted to murder", then the third heuristic fires and both verbs are used in the pattern. Both verbs are included to produce a more specific pattern, e.g. "attempted to kill" is more informative than just "attempted". In most cases, the verb is not followed by an infinitive so the active verb is used by itself, e.g., "bombed <target>". The heuristics are applied in a specific order and the first one that fires (usually the most specific one) is the one that wins.[8] Finally, if the verb is an auxiliary verb then AutoSlog recognizes that the verb does not carry any semantics itself so the head noun of the direct object is included as part of the pattern, e.g., "X was a victim".

The heuristics propose a trigger word for a concept node definition and a set of enabling conditions that must be satisfied to recognize the complete pattern. Concept node definitions also contain a slot to extract the information.[9] The heuristics specify which syntactic constituent should be used for the slot expectation. For example, if the noun phrase is identified as the subject

[7]An auxiliary verb is any form of "to be" or "to have". With respect to the AutoSlog heuristics, the term **auxiliary** implies that the main verb is a form of "to be" or "to have".

[8]The heuristics are ordered in Figure 3.2 for the sake of readability which is not the order in which they are applied. The most specific patterns are applied first.

[9]In principle, concept nodes can have multiple slots to extract multiple pieces of information. However, all of the concept nodes generated by AutoSlog have only a single slot.

of the clause then the resulting concept node is defined with a slot that expects its filler to be the subject of the clause. The name of the slot (e.g., *victim*) comes from the template slot where the information was originally found. In order to generate domain-dependent concept nodes, AutoSlog requires three domain specifications. One of these specifications is a set of mappings from template slots to concept node slots. For example, information found in the human target slot of a template maps to a *victim* slot in a concept node.

Several additional parts of a concept node definition must also be specified: hard and soft constraints for each slot, and an event type. The second set of domain specifications are hard and soft constraints for each type of concept node slot, for example semantic constraints to specify a legitimate perpetrator. In the domain of terrorism, perpetrators must have one of four semantic types: HUMAN, PROPER-NAME, TERRORIST, or ORGANIZATION.

Each concept node also has an event type. For the MUC-4 terrorism domain, in most cases, a concept node was assigned the event type of the template from which it was generated (e.g., bombing, kidnapping, etc.). However, we used special types for some concept nodes. The third set of domain specifications are mappings from template types to concept node types. In general, if the targeted information was found in a kidnapping template then AutoSlog uses "kidnapping" as the concept node type. However, for the terrorism domain we used special types for information from the perpetrator and instrument template slots because perpetrators and instruments often appear in sentences that do not describe the nature of the event (e.g., "The FMLN claimed responsibility" could refer to a bombing, kidnapping, etc.).[10]

The concept node definitions produced by AutoSlog are specific to the CIRCUS sentence analyzer. However, in theory, AutoSlog could be used in conjunction with other sentence analyzers. The concept nodes produced by AutoSlog are really just patterns for information extraction that could be adapted for other systems. To generate these patterns, AutoSlog requires a sentence analyzer that can separate raw text into clauses and identify the major syntactic constituents of each clause, i.e., subjects, verbs, direct objects, and prepositional phrases. CIRCUS has many additional functionalities, but only the syntactic recognition components of CIRCUS are used by AutoSlog.

3.4 Sample Concept Node Definitions

To illustrate how this whole process comes together, this section shows examples of concept node definitions generated by AutoSlog for the terrorism domain. Figure 3.3 shows a relatively simple concept node definition that is activated by phrases such as "was bombed", "were bombed", etc. AutoSlog created this definition in response to the input string "public buildings" which was found in the physical target slot of a bombing template from text DEV-MUC4-0657. Figure 3.3 shows the first sentence in the text that contains the string "public buildings". When CIRCUS analyzed the sentence, it identified "public buildings" as the subject of the first clause. The heuristic for the pattern **<subject> passive-verb** produced this concept node using the word "bombed" as the trigger word with enabling conditions that require a passive verb form. The concept node contains a single variable slot[11] which expects its filler to be the subject of the clause and labels it as a target because the string came from the physical target template slot.

---

[10]These domain mappings are an artifact of the templates that we used to get the targeted information. In a new domain, we would use an annotated corpus (see Section 5.4.3) instead of templates so these template mappings would be unnecessary. The only domain specifications required for an annotated corpus are the slot constraints.

[11] *Variable slots* are slots that extract information. *Constant slots* have predefined values that are used by AutoSlog to specify the concept node type.

The constraints for physical targets are pulled in from the domain specifications (described in the previous section). Finally, the concept node is given the event type *bombing* because the input string came from a bombing template.[12]

Id: DEV-MUC4-0657	**Slot filler:** "public buildings"
Sentence: In La Oroya, Junin department, in the central Peruvian mountain range, public buildings were bombed and a car-bomb was detonated.	
CONCEPT NODE	
Name:	target-subject-passive-verb-bombed
Trigger:	bombed
Variable Slots:	(target (*SUBJECT* 1))
Constraints:	(class PHYS-TARGET *SUBJECT*)
Constant Slots:	(type bombing)
Enabling Conditions:	(passive)

Figure 3.3: Concept Node For "<target> was bombed"

Figure 3.4 shows an example of a good concept node that has more complicated enabling conditions. In this case, CIRCUS found the targeted string "guerrillas" as the subject of the first clause and AutoSlog applied a different heuristic than in the previous example. The heuristic for the pattern **<subject> verb infinitive** matched the phrase "threatened to murder" and generated a concept node with the word "murder" as its trigger with enabling conditions that require the preceding words "threatened to" in an active construction. The concept node has a slot that expects its filler to be the subject of the clause and expects it to be a perpetrator (because the slot filler came from a perpetrator template slot). The constraints associated with perpetrators are incorporated and the concept node is assigned the type "perpetrator" because the domain specifications map the perpetrator template slots to perpetrator types. Note that this concept node does not extract the direct object of "threatened to murder" as a victim; a separate concept node definition is needed to pick up the victim.

Figure 3.5 shows a concept node proposed by AutoSlog that looks a bit strange. The input to AutoSlog is a group of perpetrators, "3 young individuals". AutoSlog found the "3 young individuals" in a prepositional phrase and the pp-attachment algorithm attached it to the verb "riddled". The proposed concept node recognizes phrases of the form "riddled by <perpetrator>". This expression sounds bizarre because the verb "riddled" is usually used in combination with ammunition, such as "riddled with bullets" or "riddled with gunfire". On the other hand, the verb "riddled" is not very ambiguous and, especially in a constrained corpus, is not likely to occur in other types of phrases. Therefore, although the pattern "riddled by <perpetrator>" is not the best possible pattern, it is good enough to reliably extract the appropriate kind of information.

Although the preceding definitions are clearly useful for the domain of terrorism, many of the definitions that AutoSlog generates are of dubious quality. Figure 3.6 shows an example of a bad definition. AutoSlog finds the input string, "Gilberto Molasco", as the direct object of the first clause and constructs a concept node that is triggered by the word "took" as an active verb. The concept node expects a victim as the direct object and has the event type *kidnapping*. Although

[12]Given an annotated corpus instead of templates, this information would come from the semantic tags assigned to each string by the user.

Id: DEV-MUC4-0071 **Slot filler:** "guerrillas"
Sentence: The Salvadoran guerrillas today threatened to murder individuals involved in 19 March presidential elections if they do not resign from their posts.

CONCEPT NODE

Name:	perpetrator-subject-verb-infinitive-threatened-to-murder
Trigger:	murder
Variable Slots:	(perpetrator (*SUBJECT* 1))
Constraints:	((class ORGANIZATION *SUBJECT*) (class TERRORIST *SUBJECT*) (class HUMAN *SUBJECT*) (class PROPER-NAME *SUBJECT*))
Constant Slots:	(type perpetrator)
Enabling Conditions:	((active)) (trigger-preceded-by 'threatened 'to))

Figure 3.4: Concept node for "<perpetrator> threatened to murder"

Id: DEV-MUC4-0011 **Slot filler:** "3 young individuals"
Sentence: Lopez Albujar, former army commander general and defense minister until May 1989, was riddled with bullets by 3 young individuals as he was getting out of his car in an open parking lot in a commercial center in the residential neighborhood of San Isidro.

CONCEPT NODE

Name:	perpetrator-pp-passive-verb-riddled-by
Trigger:	riddled
Variable Slots:	(perpetrator (*PREP-PHRASE* (is-prep? 'by)))
Constraints:	(class WEAPON *PREP-PHRASE*)
Constant Slots:	(type perpetrator)
Enabling Conditions:	(passive)

Figure 3.5: Concept node for "riddled by <perpetrator>"

this concept node is appropriate in this sentence because this text does describe a kidnapping, in general the system should not generate a kidnapping concept node every time it sees the word "took". Constraining the direct object to be a person is also not enough because you can take a friend to the movies, you can take a family member to visit relatives, or you can take a child to school, etc.

Id: DEV-MUC4-1192 **Slot filler:** "Gilberto Molasco"
Sentence: They took 2-year-old Gilberto Molasco, son of Patricio Rodriguez, and 17-year-old Andres Argueta, son of Emimesto Argueta.

CONCEPT NODE

Name:	victim-active-verb-dobj-took
Trigger:	took
Variable Slots:	(victim (*DOBJ* 1))
Constraints:	(class VICTIM *DOBJ*)
Constant Slots:	(type kidnapping)
Enabling Conditions:	(active)

Figure 3.6: Concept node for "took <victim>"

AutoSlog generates poor definitions for many reasons. In the previous example, AutoSlog generated the best possible pattern for the given sentence but the pattern was not reliable in general. Sometimes a sentence does contain a reliable expression but AutoSlog does not find it. Figure 3.7 shows a sentence that mentions a robbery incident where the perpetrators are a group of "soldiers with their faces painted black". AutoSlog correctly identified the soldiers as the subject of the first clause, but CIRCUS interpreted the word "painted" as an active verb. The resulting concept node represents the pattern "<perpetrator> painted"; every time the verb "painted" appears in a text the concept node will extract its subject as a perpetrator. This is clearly not a useful expression for the terrorism domain. This sentence is difficult for AutoSlog for several reasons. The second verb in the sentence, "arrived", is not useful for triggering a terrorism concept node either. Ideally, we would like AutoSlog to propose the pattern "<perpetrator> looted", "<perpetrator> burned", or perhaps even "<perpetrator> broke down". To get these patterns, however, AutoSlog would have to skip over both the verbs "painted" and "arrived". Without semantic information, AutoSlog cannot always locate the most appropriate phrases.

Figure 3.8 shows another example of a concept node definition that represents a bad pattern. AutoSlog found the targeted information, "machineguns", in a prepositional phrase and incorrectly attached it to the noun "priests". The resulting concept node definition will look for expressions of the form "priests with X" and extract X as a weapon. This pattern is amusing, but it is not reliable. If the pp-attachment algorithm had correctly attached the machineguns to the word "killing" then AutoSlog would have produced a good definition to recognize patterns of the form "killing with <weapon>".

AutoSlog generates bad definitions for many reasons, such as (a) when a sentence contains the targeted noun phrase but does not describe the event (i.e., the assumption mentioned in Section 3.2 does not hold), (b) when a heuristic proposes the wrong conceptual anchor point, for example when the pp-attachment algorithm makes a mistake, or (c) when CIRCUS incorrectly analyzes the sentence. These dubious definitions prompted us to include a human in the loop to

Id: DEV-MUC4-0058 **Slot filler:** "soldiers with their faces painted black"
Sentence: According to the report, soldiers with their faces painted black arrived in Cayara last Saturday and broke down doors, looted stores, and burned several houses.

CONCEPT NODE

Name:	perpetrator-subject-active-verb-painted
Trigger:	painted
Variable Slots:	(perpetrator (*SUBJECT* 1))
Constraints:	((class ORGANIZATION *SUBJECT*) (class TERRORIST *SUBJECT*) (class PROPER-NAME *SUBJECT*) (class HUMAN *SUBJECT*)
Constant Slots:	(type PERPETRATOR)
Enabling Conditions:	(active)

Figure 3.7: Concept node for "<perpetrator> painted"

Id: DEV-MUC4-0826 **Slot filler:** "machineguns"
Sentence: Ambassador William Walker, if you still have any shame, tell the world and answer this question: if the armed forces general staff did not kill the jesuit priests, how could the murderers – as this international dispatch says – remain in the residence for 1 hour after the heavy shooting, after killing the priests with machineguns in tripods, as the cable says?

CONCEPT NODE

Name:	instrument-pp-noun-priests-with
Trigger:	priests
Variable Slots:	(instrument (*PREP-PHRASE* (pp-check 'with)))
Constraints:	(class WEAPON *PREP-PHRASE*)
Constant Slots:	(type weapon)
Enabling Conditions:	(noun-triggered)

Figure 3.8: Concept node for "priests with <instrument>"

weed out bad concept node definitions.[13] In the next section, we explain the evaluation procedure and present empirical results for AutoSlog in three domains.

3.5 Experimental Results

3.5.1 The Terrorism Domain

To evaluate AutoSlog, we used AutoSlog to create a dictionary for the MUC-4 domain of terrorism and compared it with the concept node dictionary that that was hand-crafted for MUC-4. As training data, we used the 1500 texts and their associated answer keys from the MUC-4 development corpus, which contained 772 relevant texts. AutoSlog only uses the relevant texts. The input to AutoSlog was the set of slot fillers from six MUC-4 template slots that contained string fills because these noun phrases could be easily mapped back to the source text. These six slots contained the following types of information:

Table 3.1: Targeted information for the terrorism domain

Slot Name	Description	Example
human target description	description of victim	"a security guard"
human target name	name of victim	"Ricardo Castellar"
instrument id	weapon	"car-bomb"
perpetrator individual	name or description of people	"a group of subversives"
perpetrator organization	name of organization	"the FMLN"
physical target id	description of physical target	"car dealership"

The 1258 answer keys for these 772 texts contained 4780 string fills which were given to AutoSlog as input along with their corresponding texts.[14] In response to these strings, AutoSlog generated 1237 unique concept node definitions. AutoSlog does not necessarily generate a definition for every input string, for example when no heuristic applies or when sentence analysis goes awry. Also, AutoSlog is smart enough to refrain from generating duplicate definitions. For example, many texts contain expressions of the form "X was kidnapped" so AutoSlog proposed a concept node definition for this pattern in response to many different input strings. AutoSlog keeps track of the definitions that it proposes and will not generate the same definition twice. Instead, it keeps a count associated with each definition that indicates how many times the definition was proposed by AutoSlog. For example, AutoSlog proposed a concept node to recognize the pattern "X was kidnapped" 46 times in response to the 4780 input strings. Table 3.2 shows the patterns that are recognized by concept nodes that were proposed at least 25 times by AutoSlog.

Not surprisingly, the patterns shown in Table 3.2 represent expressions that are common in texts describing terrorism. The patterns most frequently proposed by AutoSlog are likely to be

[13]In Section 3.6.1 we show that an unfiltered dictionary for the terrorism domain performs substantially worse than filtered dictionaries.

[14]Many of the slots contained several possible strings ("disjuncts"), any one of which is a legitimate filler. In this case, AutoSlog identified the first sentence that contained any of the strings.

Table 3.2: Frequently proposed patterns for terrorism

Linguistic Pattern	Number of Times Proposed
<victim> was killed	121
murder of <victim>	111
assassination of <victim>	95
<victim> was wounded	50
<victim> was kidnapped	46
<weapon> exploded	43
killed <victim>	42
death of <victim>	40
murdered <victim>	36
<victim> died	35
<victim> was murdered	34
<perpetrator> attacked	32
<victim> was injured	29
<victim> was assassinated	29
kidnapped <victim>	29
killing <victim>	28
members of <perpetrator>	25

the ones that are most important for the domain. However, note that the patterns most frequently proposed by AutoSlog are usually *important* for the domain but not always *exclusive* to the domain. For example, the concept node most frequently proposed by AutoSlog represents the expression "<victim> was killed". This concept node is crucial for the domain of terrorism because people are often killed in terrorist incidents. If this expression was not in the dictionary, the system would fail to extract many victims of terrorism. However, people are also killed in many ways that have nothing to do with terrorism. Therefore this expression will also appear in many texts that do not mention terrorism. AutoSlog's job is not necessarily to find patterns that are exclusive to the domain but to find patterns that are useful for the domain.

AutoSlog does not make use of the frequency counts associated with the concept nodes, but Section 3.6.2 describes how they were used in an experiment with government analysts. To give some idea of the magnitude of duplicate suppression, consider that, if we include duplicates, AutoSlog actually proposed 3860 concept node definitions in response to the 4780 input strings. This implies that, on average, AutoSlog proposed each definition three times. However, the distribution is highly skewed. Figure 3.9 displays a histogram showing the frequency distribution of the concept node definitions for the terrorism domain. For example, 680 different concept nodes were proposed exactly once. At the other end of the spectrum, one concept node was proposed 121 times ("<victim> was killed").

As we mentioned in Section 3.4, not all of the concept node definitions proposed by AutoSlog are good ones. Therefore we put a human in the loop to filter out definitions that might cause trouble. For this experiment, the user was a second-year graduate student who had some experience with the MUC-4 system and AutoSlog but was not one of the original system developers.[15] An interface displayed each dictionary definition proposed by AutoSlog and asked him to put each

[15]In Section 3.6 we show that novice users can also achieve good results with AutoSlog.

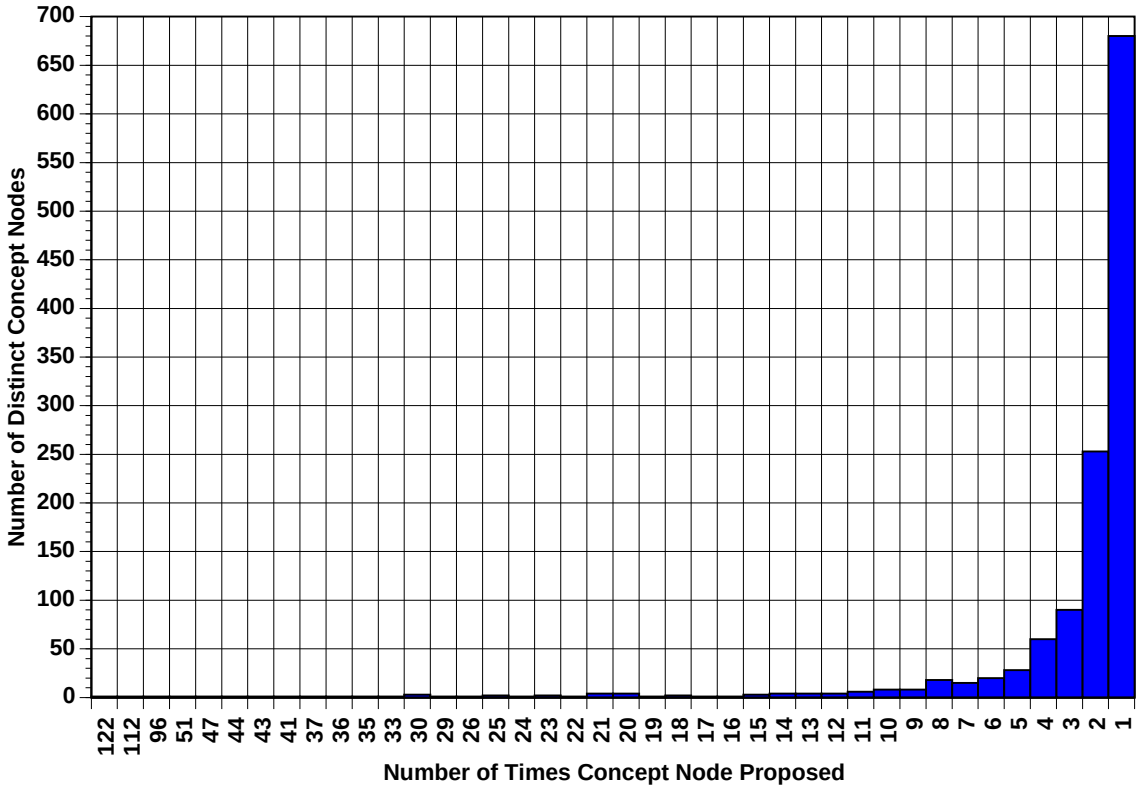


Figure 3.9: Histogram of concept node frequencies in the terrorism domain

definition into one of two piles: the "keeps" or the "rejects". The "keeps" were good definitions that could be added to the permanent terrorism dictionary without alteration.[16] The "rejects" were definitions that required additional editing to be salvaged, were obviously bad, or were of questionable value. It took the user 5 hours to sift through the 1237 unique definitions proposed by AutoSlog. The filtered dictionary contained 450 definitions (only the "keeps"), which we used as our final concept node dictionary. The number of definitions kept by the user for each of the six types of concept nodes is shown in Table 3.3.

The percentage of definitions that were kept differs widely across the different slots. For example, 72% of the instrument definitions were retained by the user but only 18% of the human target description definitions were kept. To explain why the slots have different retention rates, we must describe some of the issues that are involved in the filtering process.

1. **Is the concept node likely to appear in relevant texts?**

   Ideally, a good concept node definition represents an expression that is common in relevant texts and but uncommon in irrelevant texts. However, many expressions that are common in relevant texts are also common in irrelevant texts. Therefore the user must only focus on whether the pattern is likely to appear in relevant texts. The crucial question is the following: if this expression is not in the dictionary, will the system miss a lot of relevant information? If so, then the user should keep the concept node. As we just explained, a good example of

[16]The only exception is that the user could change the event type if that was the only revision needed.

Table 3.3: AutoSlog dictionary for terrorism

Slot Name	#CNs Kept	#CNs Proposed	%CNs Kept
human target description	34	191	18%
human target name	51	169	30%
instrument id	93	129	72%
perpetrator individual	102	303	34%
perpetrator organization	31	165	19%
physical target id	139	280	50%
TOTAL	450	1237	36%

this phenomenon occurs with the expression "<victim> was killed". In the MUC-4 corpus, many people are killed in military incidents that are not relevant to the terrorism domain. However, many people are also killed in terrorist events. If this expression is not in the dictionary, the system will fail to extract many victims of terrorism. Therefore the user should keep this concept node even though it will also extract many irrelevant victims.[17]

2. **Does this concept node represent an event type?**

   Some concept node patterns represent event types but others do not. For example, the pattern "<victim> was kidnapped" refers to a kidnapping event and the pattern "<target> was bombed" refers to a bombing event. In the MUC-4 corpus, we found that many of the expressions that refer to victims and targets identify the type of event. However, instruments and perpetrators often appear in expressions that do not identify the event type. For example, the phrase "three men were arrested" does not indicate whether the men were arrested for a bombing, murder, kidnapping, or shoplifting incident. Similarly, the phrase "an M-16 rifle was seized" does not indicate whether the rifle was seized during an attack or a drug raid. Victims and targets are usually reported in the first few sentences of a news article, but instruments and perpetrators are often reported later in the article where the event is not mentioned explicitly but is only evident from context.

   Because of this phenomenon, the victim and physical target concept nodes were labeled with event types[18] (if applicable) but the instrument and perpetrator concept nodes were not. The event type labels were used by the UMass/MUC-4 discourse analyzer to determine whether a template should be generated for a text. If none of the concept nodes generated by a text had event type labels, then the discourse analyzer assumed that the story did not mention a relevant event so did not generate a template. The human-in-the-loop who filtered the terrorism dictionary knew how the discourse analyzer worked, so he was more liberal about keeping the instrument and perpetrator concept nodes. That is, he knew that spurious instrument and perpetrator concept nodes would not cause a template to be

[17] In a real application, the job of sifting through the relevant and irrelevant information extracted by the system is usually handled later by a discourse analysis module.

[18] Assigned automatically from the key template, although the user could modify it.

generated unless there was other evidence in the text that identified a relevant event.[19] The fact that the user kept 72% of the instrument definitions implies that he took advantage of this knowledge and adopted a liberal filtering strategy for the instrument definitions.[20]

3. **Ordering effects**

   AutoSlog created concept nodes based on six different types of information from the templates (shown in Table 3.1). However, AutoSlog created only four classes of concept nodes. Since the *human target description* slot and the *human target name* slot both contained references to victims, we set up the domain mappings so that AutoSlog created a general *human target* concept node for both types of input. Similarly, the *perpetrator individual* slot and the *perpetrator organization* slot both contained references to perpetrators so we set up the domain mappings so that AutoSlog created a general *perpetrator* concept node for both. Since AutoSlog will not generate duplicate concept node definitions, it generally proposes fewer new definitions as it processes more data. The most common patterns are encountered first so AutoSlog proposes many definitions initially, but as AutoSlog processes more data it repeatedly encounters many of the same patterns so few new definitions are produced. We presented AutoSlog with the *human target name* data before the *human target description* data (and likewise the *perpetrator individual* data before the *perpetrator organization* data), so the most common patterns for victims were found during the acquisition phase for the *human target name*. By the time AutoSlog got to the *human target description*, most of the common patterns had already been acquired. Therefore we expect a greater proportion of the *human target description* definitions to represent bad patterns that will be rejected during the human-in-the-loop filtering process. As you can see in Table 3.3, both the *human target description* data and the *perpetrator organization* data produced a lower percentage of good definitions than the others.

Finally, we compared the filtered AutoSlog concept node dictionary[21] with the hand-crafted MUC-4 dictionary. To ensure a clean comparison, we tested the AutoSlog dictionary using the official UMass/MUC-4 system. The resulting "AutoSlog" system was identical to the UMass/MUC-4 system except that we replaced the hand-crafted concept node dictionary with the AutoSlog dictionary. We evaluated both systems on the basis of two blind test sets of 100 texts each. These were the TST3 and TST4 texts that were used in the final MUC-4 evaluation. We then

[19]However, these concept nodes may extract irrelevant perpetrators and instruments in *relevant* texts which can cause confusion during discourse analysis.

[20]In this case, the human-in-the-loop had knowledge about how the concept nodes would ultimately be used by the information extraction system. To be fair, we also gave this information to the students and analysts who filtered terrorism dictionaries in the experiments described in Sections 3.6.1 and 3.6.2. In general, any knowledge about how the concept nodes will be used can be exploited during the filtering process.

[21]We augmented the AutoSlog dictionary with 4 meta-level concept nodes from the hand-crafted dictionary before the final evaluation. These are special concept nodes that just recognize textual cues for discourse analysis and do not extract any slot fillers. These concept nodes are truly a separate species and we knew a priori that AutoSlog was not designed to create concept nodes for discourse cues.

scored the output generated by both systems using the MUC-4 scoring program. The results for the two systems are shown in Table 3.4.[22]

Recall refers to the percentage of the correct answers that the system successfully extracted and *precision* refers to the percentage of answers extracted by the system that were actually correct. The *F-measure* is a single measure that combines recall and precision, in this case with equal weighting. The formula for the F-measure is:

$$F(\beta) = \frac{(\beta^2 + 1.0) \times P \times R}{\beta^2 \times P + R}$$

where P is precision and R is recall. These are all standard measures used in the information retrieval community that were adopted for the final MUC-4 evaluation [MUC-4 Proceedings, 1992].

Table 3.4: Comparative results

System/Test Set	Recall	Precision	F-measure
MUC-4/TST3	46	56	50.51
AutoSlog/TST3	43	56	48.65
MUC-4/TST4	44	40	41.90
AutoSlog/TST4	39	45	41.79

The UMass/MUC-4 system was among the top-performing systems in MUC-4 [Lehnert *et al.*, 1992b] so the results in Table 3.4 were roughly state-of-the-art. Table 3.4 shows that the AutoSlog dictionary achieved almost the same level of performance as the hand-crafted dictionary on both test sets. Comparing F-measures, we see that the AutoSlog dictionary achieved 96.3% of the performance of our hand-crafted dictionary on TST3, and 99.7% of the performance of the official MUC-4 system on TST4. For TST4, the F-measures were virtually indistinguishable and the AutoSlog dictionary achieved better precision than the original hand-crafted dictionary. We should also mention that we augmented the hand-crafted dictionary with 76 concept nodes created by AutoSlog before the final MUC-4 evaluation. These definitions improved the performance of our official system by filling gaps in its coverage. Without these additional concept nodes, the AutoSlog dictionary would likely have shown even better performance relative to the MUC-4 dictionary.

So far, we have shown that AutoSlog can generate useful definitions for the domain of terrorism. But will AutoSlog's heuristics generalize to other domains? Are additional heuristics needed? To answer these questions, we used AutoSlog to construct dictionaries for two additional domains: joint ventures and microelectronics. The next two sections describe our experience with AutoSlog in these domains.

3.5.2 The Joint Ventures Domain

For the second set of experiments, we used AutoSlog to build a dictionary for the MUC-5 domain of joint venture activities, which is described in detail in Section 2.3.1. For the JV domain, AutoSlog created concept nodes to extract eight different types of information, shown in Table 3.5.

---

[22]The results in Table 3.4 do not correspond to our official MUC-4 results because we used "batch" scoring and an improved version of the scoring program for the experiments described here.

Table 3.5: Targeted information for the joint ventures domain

Slot Name	Description	Example
entity name	company, govt., or person	"Toyota Motor Corp."
facility name	name of facility	"Beijing jeep plant"
ownership percent	percent of ownership	"51%"
ownership total capitalization	total capitalization amount	"$46,000,000"
person name	name of person	"Paul Phillips"
product/service	industry product or service	"V2500 jet engine"
revenue rate	expected revenue rate	"$80,000,000 per year"
revenue total	total expected revenue	"$80,000,000"

Although these information types are general in nature, the MUC-5 systems were only supposed to extract information pertaining to joint venture activities. For example, the systems were supposed to extract the names of companies participating in a joint venture, facilities used by a joint venture company, ownership percents associated with a joint venture company, etc.

The joint venture domain is similar to the terrorism domain in the sense that they are both event-driven. The terrorism texts revolve around incidents such as bombings, murders, and kidnappings. The JV texts revolve around joint venture activities, such as two companies forming or dissolving a joint venture. Because of the similar nature of the domains, we expected AutoSlog to do well with the JV domain.

The goals for this experiment were twofold. The first goal was to evaluate the generality of AutoSlog across different domains. In particular, we wanted to determine whether AutoSlog's heuristics were general enough to produce useful concept node definitions for domains other than terrorism. The second goal was to learn something about what types of domains are appropriate for AutoSlog and what types of domains are not. However, it is important to keep in mind that we also relied on AutoSlog to create a JV dictionary for MUC-5; we did not have a hand-crafted dictionary to fall back on. So we first applied the original set of AutoSlog heuristics to the joint venture texts, but we were willing to revise, delete, or add new heuristics if we saw the need to do so.

3.5.2.1 Moving AutoSlog to a New Domain

During the months preceding MUC-5, we made several changes to AutoSlog. In the end, however, we were surprised by how little had actually changed. The original set of heuristics remained largely intact and most of the changes to AutoSlog were small. The final set of AutoSlog heuristics used for the joint ventures domain are shown in Table 3.10.

We made three changes to the original set of AutoSlog heuristics. We added two new patterns, **<subject> verb direct-object** and **infinitive preposition <noun-phrase>**, and we removed one heuristic, **passive-verb <dobj>**. We added the **infinitive preposition <noun-phrase>** heuristic to represent patterns such as "to collaborate on a project". We simply hadn't seen this pattern very often in the terrorism domain, probably because terrorist events are usually reported in the past tense whereas joint venture activities are often reported in the future tense (e.g., "Companies X and Y will be cooperating ..."). Second, we dropped the **passive-verb <dobj>** heuristic. This heuristic was in the terrorism system only because, in its early stages, CIRCUS had trouble distinguishing active and passive verb form. In principle, this heuristic should never have fired anyway unless CIRCUS made a mistake.

Therefore, the only major change to the set of heuristics is the new pattern, **<subject> verb direct-object**, which represents expressions such as "Toyota and Nissan formed a joint

Linguistic Pattern	Example
<subject> passive-verb	<entity> was formed
<subject> active-verb	<entity> linked
<subject> verb direct-object	<entity> completed acquisition
<subject> verb infinitive	<entity> agreed to form
<subject> auxiliary noun	<entity> is conglomerate
active-verb <dobj>	acquire <entity>
infinitive <dobj>	to acquire <entity>
verb infinitive <dobj>	agreed to establish <entity>
gerund <dobj>	producing <product-service>
noun auxiliary <dobj>	partner is <entity>
noun preposition <noun-phrase>	partnership between <entity>
active-verb preposition <noun-phrase>	buy into <entity>
passive-verb preposition <noun-phrase>	was signed between <entity>
infinitive preposition <noun-phrase>	to collaborate on <product-service>

Figure 3.10: AutoSlog patterns for the joint ventures domain

venture". We found an important difference between the language typically used to describe terrorist events and the language used to describe joint venture activities. In the terrorism domain, verbs usually represent the semantics describing the event. For example, the words "bombed, "murdered", and "kidnapped", carry the semantic information corresponding to the event types. In the joint ventures domain, verbs are much weaker. Even though it is an event-driven domain, the nouns typically contain the most important semantic information. For example, joint ventures are often reported using expressions such as "X and Y formed a joint venture", "X completed an acquisition", "X signed an agreement", etc. The verbs alone ("formed", "completed", and "signed") do not necessarily describe joint venture activities. The verbs in combination with the nouns (e.g., "venture", "acquisition", and "agreement") describe specific joint venture activities.

The original **<subject> active-verb** heuristic would have proposed concept nodes to recognize expressions such as "X formed", "X completed", and "X signed". Since these expressions are too general for the JV domain, we added the new heuristic that includes the head noun of the direct object when one is available. This heuristic takes precedence over the old one, so if a direct object is present then its head noun is used in the concept node pattern. If a direct object is not present, then the new heuristic fails and AutoSlog falls back on the original heuristic. The new heuristic produced many useful concept nodes that recognized expressions such as "X formed venture", "X completed acquisition", and "X signed agreement".

We made a few other changes to AutoSlog as well. In the JV domain, particles occur in many important expressions, for example: "set up venture", teamed up with", "linked up with", and "carrying out study". The terrorism domain also includes important expressions involving particles (such as "blew up", "blown up", "carried out"). We used a manually crafted phrasal lexicon to recognize these expressions in the UMass/MUC-4 terrorism system so they were treated as single verbs, such as "blew_up", "blown_up", "carried_out".

Since particles are important in the JV domain, we gave AutoSlog the ability to recognize particles. For all of the heuristics involving verbs, the new version of AutoSlog looks for particles immediately following the verb. If a particle is found, then it is included in the pattern. For example, given the sentence "Company X was set up by ...", the **<subject> passive-verb** heuristic

fires, AutoSlog finds the particle "up" following the verb "set", so the proposed concept node represents the pattern "<entity> was set up". In contrast, the old version would have proposed the pattern "<entity> was set", which is not as reliable. In retrospect, including particle recognition would have been useful in the terrorism domain as well. The improved heuristics would have automatically created patterns for many of the phrases that we manually encoded in the phrasal lexicon for terrorism.

Numerical values are also important in the JV domain, whereas they are not relevant to terrorism. In particular, ownership percentages and monetary values are crucial. The UMass/MUC-5 system includes special-purpose functions ("specialists") to recognize percentages and monetary objects, e.g., 51% and $50,000,000. When we ran the original AutoSlog heuristics through the JV corpus, AutoSlog proposed concept nodes for overly specific patterns, such as "<entity> controls 51%", "<entity> owns 49%", "<entity> invested $50000000". There is nothing wrong with these patterns except that they provide no generality. For example, if a company controls 50% or 52% in a future text, then the company will not be extracted.

To allow AutoSlog to generalize over specific values, we gave AutoSlog access to the specialists. In the new version of AutoSlog, the heuristics check to see whether a noun is a percentage or monetary object. If so, then AutoSlog generalizes the pattern to represent all objects of the same type. For example, given the sentence "IBM controls 51% of ...", the **<subject> verb direct-object** heuristic fires, AutoSlog finds a percentage object as the head noun of the direct object and proposes a concept node that recognizes the pattern "<entity> controls PERCENT-OBJECT". This concept node will be activated by all expressions of the form "<entity> controls X%". Similarly, the new version of AutoSlog created a concept node to recognize expressions of the form "<entity> will invest MONETARY-OBJECT".

For the sake of completeness, we will briefly mention a few other improvements that were added to AutoSlog. We replaced the pp-attachment algorithm with a new frequency-based pp-attachment algorithm, which is described in the next section. We divided the heuristics involving auxiliary verbs (**<subject> auxiliary noun** and **noun auxiliary <dobj>**) into separate heuristics that distinguish between forms of the verb "to be" and "to have". And we decided to treat communication verbs, such as "said", "reported" and "announced", as a special case. These verbs are meta-level verbs that do not carry any semantic content on their own so the new version of AutoSlog skips over clauses containing these verbs. These changes to AutoSlog were all general improvements that would probably have improved the terrorism system as well.[23]

3.5.2.2 A Frequency-Based PP-attachment algorithm

As we described in Section 3.3, the original version of AutoSlog used a simple pp-attachment algorithm that made a lot of mistakes. When a prepositional phrase is attached incorrectly, the resulting concept node definition usually doesn't make much sense and is eventually thrown away during manual filtering. Consequently, errors by the pp-attachment algorithm cause two problems:

1. An incorrect attachment is usually a missed opportunity for a good concept node. This is not necessarily a major problem because common patterns occur many times in a corpus so AutoSlog has multiple opportunities to create concept nodes for them. However, AutoSlog may miss opportunities to create good concept nodes for less common patterns that occur only once or twice.
2. Most incorrect attachments result in poor concept nodes that need to be filtered by a human. Each bad concept node increases the amount of time required for the human-in-the-loop to filter the dictionary.

[23]However, treating communication verbs as a special case might not generalize to genres other than news reports.

We did not formally evaluate the original pp-attachment algorithm, but a lot of the bad concept nodes produced by AutoSlog were the result of incorrect attachments. So we replaced the original pp-attachment algorithm with a new frequency-based algorithm that uses collocation data derived from a training corpus to make prepositional phrase attachment decisions.[24]

The motivation for this algorithm comes from the tendency of prepositional phrases toward right association; that is, prepositional phrases often attach to the most recent noun phrase or verb phrase. Although this is not always the case, the idea behind our approach is that some attachment preferences will become apparent from collocation data collected over a training corpus. For example, in the MUC-5 joint ventures corpus the word "venture" immediately precedes the preposition "with" 270 times. This high frequency collocation implies that the word "venture" is strongly associated with the word "with". It follows that a prepositional phrase beginning with the word "with" is likely to attach to a preceding noun phrase with the head noun "venture".

The statistical pp-attachment algorithm involves three steps:

1. **Preprocess the Corpus:** Preprocess the texts to identify separate sentences, normalize date expressions, etc.
2. **Collect Collocation Data:** For each preposition, determine which words *immediately* precede the preposition in the corpus and how often each word precedes the preposition.
3. **Resolve Attachment:** Given a prepositional phrase that needs to be attached, generate all possible attachment points.[25] For each possible attachment point, find the frequency of the collocation between the verb or head noun of the constituent and the preposition. Choose the constituent with the highest frequency collocation as the attachment point. If there is a tie then break the tie by choosing the closest constituent.[26]

As an example, consider the following sentence:

Last month Sime entered into joint ventures with the Singapore-based Sembawang Shipyard for a fabrication plant in Johore and with Nikkon Kokkan of Japan to tender for oil and gas engineering projects.

To attach the prepositional phrase "with Nikkon Kokkan", the following collocation profile of possible attachment points was generated:

entered (0), ventures (26), Shipyard (0), plant (16), Johore (0)

The words "entered", "Shipyard", and "Johore" never immediately preceded the preposition "with" in the training corpus. The statistical approach has a tendency to eliminate proper nouns, numbers, and very specific words from consideration because each one appears infrequently (if at all) in the training corpus. The words "ventures" and "plant" were collocated with the preposition "with" 26

[24]Theoretically, we believe that semantic information is necessary to make many pp-attachment decisions correctly. However, for practical purposes, we wanted AutoSlog to remain domain-independent so we did not want the pp-attachment algorithm to depend on any additional resources.

[25]We included the previous verb, direct object, and intervening prepositional phrases if they were present.

[26]If all frequency collocations are zero then we apply the original pp-attachment algorithm instead of choosing the closest constituent.

times and 16 times, respectively. Even though "ventures" is farther away, the algorithm chooses "ventures" as the most likely attachment point because it has a higher collocation value.

We generated collocation data for nine prepositions (at, between, by, for, from, in, on, to, with)[27] for both the joint ventures corpus and the microelectronics corpus. Appendix 7.3 shows the words with highest collocation frequencies for four prepositions in each domain. We compared the new pp-attachment algorithm with the old one on a small set of 100 pp-attachment decisions in the joint ventures domain.[28] In 85 of the 100 cases, both algorithms chose the same attachment point, primarily because many attachments are unambiguous or involve the preposition "of", which almost always attaches to its closest constituent. Of the 15 cases where they differed, the new algorithm was correct 12 times and the old algorithm was correct three times, although only one of the three cases was compelling.[29]

The strength of the frequency-based pp-attachment algorithm is its ability to find "heavy hitters" that have strong preferences for certain prepositions. For example, if the word "venture" is among the possible attachment points for a prepositional phrase with the preposition "with" then the algorithm will skip over anything in the middle and use "venture" as the attachment. This strategy works particularly well for domain-specific preferences. Since the collocation data was generated from a domain-specific corpus, many words that are important to the domain occur with high frequency. Since AutoSlog applies the pp-attachment algorithm only to sentences that contain domain-specific information, the algorithm is especially well-suited for identifying appropriate attachments in these sentences.

Originally, we intended to use a frequency threshold so that this algorithm would not be applied if all of the possible attachment points had low frequencies, under the assumption that low frequencies would not be reliable. However, we found that the algorithm often made good decisions even with relatively low frequencies. In particular, proper names, geographic locations, and numbers often have very low frequencies because each individual name, location, or number may occur only once or twice (if at all) in the training corpus. Therefore, the algorithm has a tendency to prefer almost any other word as an attachment point as long as its frequency is at least a little bit higher. In practice, names, locations and numbers are rarely good attachment points so the algorithm is almost always correct to rule these out. The algorithm also works well in the face of errors by the sentence analyzer. When CIRCUS incorrectly tags a word as a noun or verb, that word often has a very low or zero collocation frequency so the pp-attachment algorithm usually chooses another attachment point with at least a slightly higher frequency. In short, it was impressive to see the algorithm automatically skipping over proper nouns, numbers, and errors produced by CIRCUS. Therefore, although low collocation frequencies are generally not reliable, the algorithm performed surprisingly well with low frequencies so we decided not to bother picking an arbitrary frequency cutoff.

This pp-attachment algorithm is far from perfect and has several problems. In particular, verbs that always take direct objects will *never* be adjacent to prepositions and the collocation statistics for the preposition "to" are confounded by the cases where "to" is an infinitive. However, this statistical approach represents a domain-independent algorithm for making pp-attachment

---

[27]We specifically did not apply this algorithm to the preposition "of" because "of" almost always attaches to its closest preceding constituent.

[28]We used the first 100 pp-attachment decisions that AutoSlog made while generating concept node definitions. Remember that the pp-attachment algorithm is applied only when AutoSlog needs to decide which word should trigger a concept node.

[29]PP-attachment is a somewhat artificial task and there is not always one attachment that is obviously the best.

decisions using a training corpus. Furthermore, this algorithm is particularly well-suited for domain-specific applications where the algorithm is not required to attach every prepositional phrase in a text but only the ones that appear in domain-specific contexts.

3.5.2.3 Sample Concept Node Definitions for JV

Before we present the results for the JV dictionary, we must discuss the concept node specifications for JV (described in Section 2.1). Ideally, the dictionary definitions should extract information only in the context of a joint venture. However, this is not always possible. Most of the information described in Table 3.5 frequently appears in sentences that do not explicitly mention a joint venture. Therefore the concept node patterns are usually general and will appear in irrelevant as well as relevant texts. In fact, only the entities commonly appear in phrases that explicitly mention a joint venture, such as "Toyota Motor Corp. formed a joint venture with Nissan.". The remaining types of information typically appear in subsequent phrases or sentences that do not explicitly mention the joint venture, e.g., "The factory will produce cars" or "Sales are projected at $4000000.". Many of the entities also appear in sentences that do not mention a joint venture, e.g., "The company will be called ABC Corp.".

To distinguish expressions that refer to joint venture activities from those that do not, the JV entity concept node definitions had a *relationship* slot (similar to the event-type slot in the terrorism concept nodes). Concept nodes that represent expressions referring to joint venture activities (e.g., "<entity> formed venture" or "tie-up with <entity>") were given one of three relationship values: *jv-parent*, *jv-child*, or *jv*. *Jv-parent* indicates that the pattern extracts one of the partner entities (e.g., <entity> formed venture), *jv-child* indicates that the pattern extracts the entity formed by the joint venture, and *jv* indicates that the pattern refers to a joint venture but could extract either a partner or child entity.

Some entity concept nodes represent general patterns that extract company names but do not necessarily refer to a joint venture (e.g. "executives from <entity>") . In this case, the relationship slot was left blank. An empty relationship slot indicates that the concept node will extract entities but the entities may or may not be involved in a joint venture. Similarly, the other types of information (facilities, revenue amounts, ownership percentages, etc.) rarely occur in direct reference to a joint venture so the expressions that are useful for extracting this information usually do not refer to a joint venture. Therefore the concept nodes will pick up this information regardless of the surrounding context.[30]

The relationship slot is initially filled with a value that comes from the answer key. For example, if Company X was found in a JV template as a partner company, then the concept node definition that is proposed by AutoSlog to extract Company X is given the relationship value *jv-partner*. During the human-in-the-loop filtering process, however, the user can change this value.

Figure 3.11 shows an example of a good concept node definition proposed by AutoSlog for the joint ventures domain. The targeted information is a company, "Berliner Bank", which AutoSlog finds as the subject of the first clause. The new **<subject> verb direct-object** heuristic kicks in and AutoSlog generates a concept node that recognizes the pattern "<entity> formed venture". In the future, whenever a text contains the verb "formed" followed by a direct-object with "venture" as its head noun, this concept node will fire and extract the subject of "formed" as a jv-entity. In this example, the original AutoSlog heuristic **<subject> active-verb** would have created a concept node that was too general (i.e., "<entity> formed"). Note that the concept node has a type slot which specifies that the extracted noun phrase is probably a company, and a relationship slot which specifies that the extracted noun phrase is probably a partner. In general, this pattern

[30]In a larger application, subsequent processing modules (i.e., discourse analysis) are responsible for determining whether the extracted information relates to a relevant joint venture or not.

could also extract governments and people involved in a joint venture, not just companies, so the human-in-the-loop should change the type slot so that it does not predict only a company.

Id: 0225 **Slot filler:** "Berliner Bank"
Sentence: Berliner Bank last year formed a joint venture with KFTCIC to channel investment into medium-sized German companies.

CONCEPT NODE

Name:	jv-entity-subject-verb-and-dobj-formed-venture
Trigger:	venture
Variable Slots:	(name (*SUBJECT* 1))
Constraints:	(class JV-ENTITY *SUBJECT*)
Constant Slots:	(type jv-entity subtype company relationship jv-parent)
Enabling Conditions:	(dobj-preceded-by-verb 'formed 'venture)

Figure 3.11: Concept node for "<entity> formed venture"

Figure 3.12 shows another good concept node definition generated by AutoSlog. AutoSlog found a company called "Saft S.A." in a prepositional phrase and attached it to the verb "teamed". The resulting concept node recognizes patterns of the form "teamed up with <entity>". This definition shows how particles are included in patterns; the pattern "teamed with <entity>" would have been reasonable but "teamed up with <entity> is better.

Id: 0016 **Slot filler:** "Saft S.A."
Sentence: Japan Storage Battery Co. announced it has teamed up with a leading French battery maker, Saft S.A., to set up a joint venture in Japan to market small batteries.

CONCEPT NODE

Name:	jv-entity-pp-active-verb-teamed-up-with
Trigger:	teamed
Variable Slots:	(name (*PREP-PHRASE* (is-prep? '(with))))
Constraints:	(class JV-ENTITY *PREP-PHRASE*)
Constant Slots:	(type jv-entity subtype company relationship jv-parent)
Enabling Conditions:	((active) (particle-follows-verb 'teamed 'up))

Figure 3.12: Concept node for "teamed up with <entity>"

Figure 3.13 shows a concept node that represents the pattern "PERCENT-OBJECT by <entity>". This concept node is activated by all percentage objects and extracts an entity from

prepositional phrases that follow the percentage with the preposition "by", for example "51% by Toyota", "49% by Disney", etc. Intuitively, this pattern seems too general because it extracts information following percentages without regard to preceding context. However, in a constrained domain, this pattern is reasonably reliable. Many joint venture texts contain sentences of the form: "51% of the new company is owned by X, 25% by Y, and 24% by Z". Since this pattern is very common in the JV domain, the human-in-the-loop chose to keep this definition in the dictionary. However, the type slot should be changed because this pattern could also extract companies and governments, not just people; "person" was used as the default because the targeted information that produced this definition, "Winaryo Sulistyo", was a person.

Id: 0143	**Slot filler:** "Winaryo Sulistyo"
Sentence: Capitalized at $3000000, the first overseas manufacturing foothold of Achilles will be 40% owned by Achilles, 40% by Winaryo Sulistyo, the local investor, and 20% by Mitsubishi.	
CONCEPT NODE	
Name:	jv-entity-pp-noun-PERCENT-OBJECT-by
Trigger:	PERCENT-OBJECT
Variable Slots:	(entity (*PREP-PHRASE* (pp-check '(by))))
Constraints:	(class JV-ENTITY *PREP-PHRASE*)
Constant Slots:	(type jv-entity subtype person relationship jv-parent)
Enabling Conditions:	(noun-triggered)

Figure 3.13: Concept node for "PERCENT-OBJECT by <entity>"

Figure 3.14 shows a concept node that represents the pattern "to make <product>". Given the string "glycol" as input for a product description, AutoSlog identified "glycol" as the direct object of the verb "make". The pattern "to make <product>" seems like a good one, but the verb "make" is very general and is likely to appear in relevant as well as irrelevant texts. Even so the pattern is likely to appear in many relevant texts. Therefore we must keep it in the dictionary or the system will fail to extract important product names. The subtype slot is filled with the value "production" to denote that the concept node will usually extract objects that are produced. The relationship and subtype slots for the joint ventures domain appear in Appendix 7.3.

Finally, as we explained in Section 3.4, sometimes AutoSlog generates definitions that are bizarre and have no relevance to the domain. In Figure 3.15, AutoSlog generated a definition to recognize the pattern "<entity> thrown hat". This example shows how metaphor ("ICI has thrown its hat into the ring") can result in strange definitions. In general, it is difficult to avoid extracting information from sentences that contain metaphorical expressions because context is needed to reliably identify the metaphor.

Figure 3.16 shows a strange concept node that represents patterns of the form "fins with <entity>". This is another example of a mistake by the pp-attachment algorithm. AutoSlog found the targeted information, "Aluminium Co. of Malaysia Bhd.", in a prepositional phrase and attached it to the noun "fins". The pp-attachment algorithm should have attached it to the noun "venture" which would have produced a more sensible definition for the pattern "venture with <entity>". This is a tricky attachment because "venture" is far away from "with" and there are several other intervening verbs and nouns.

Id: 0138 **Slot filler:** "glycol"

Sentence: Mitsui and Co. said Tuesday it has reached an agreement with Union Carbide Chemicals and Plastics Co. of United States and two Taiwanese firms to set up a new firm in Canada to make ethylene glycol, a major material for polyester fibers.

CONCEPT NODE

Name:	jv-prod_serv-description-dobj-infinitive-to-make
Trigger:	make
Variable Slots:	(description (*DOBJ* 1))
Constraints:	(class JV-PROD_SERV *DOBJ*)
Constant Slots:	(type jv-prod_serv subtype production)
Enabling Conditions:	((active) (trigger-preceded-by 'make 'to))

Figure 3.14: Concept node for "to make <entity>"

Id: 0238 **Slot filler:** "ICI"

Sentence: In addition to Japanese, Taiwanese and South Korean firms, ICI has thrown its hat into the ring with 350000 ton a year PTA plants in Taiwan and Thailand.

CONCEPT NODE

Name:	jv-entity-subject-verb-and-dobj-thrown-hat
Trigger:	hat
Variable Slots:	(entity (*SUBJECT* 1))
Constraints:	(class JV-ENTITY *SUBJECT*)
Constant Slots:	(type jv-entity subtype company relationship jv-parent)
Enabling Conditions:	(dobj-preceded-by-verb 'thrown 'hat)

Figure 3.15: Concept node for "<entity> thrown hat"

But this example illustrates an important point about AutoSlog. Even if AutoSlog misses a chance to generate an important definition, such as "venture with <entity>", there will usually be other opportunities. The most common expressions appear multiple times in a training corpus so AutoSlog has many opportunities to produce definitions for the most important patterns. Therefore AutoSlog is very robust in the sense that it only needs to successfully generate each definition once and, the more common the expression, the more opportunities it has to do so.

Idiosyncratic expressions, however, may occur only once or twice. If AutoSlog misses an opportunity to create a definition for an idiosyncratic expression, then it may not get another chance. On the other hand, if an expression is not very common in the training corpus then it is not likely to be common in future texts.[31] Consequently, the absence of the definition will probably not have much impact on the final performance of the system.

Id: 0105 **Slot filler:** "Aluminium Co. of Malaysia Bhd."
Sentence: Nippon Light Metal Co. has launched a joint venture in Malaysia to produce and sell aluminum precoated fins, with Aluminium Co. of Malaysia Bhd. (ALCOM), a subsidiary of Alcan Aluminum Ltd. of Canada, Nippon Light Metal announced Wednesday.

CONCEPT NODE

Name:	jv-entity-pp-noun-fins-with
Trigger:	fins
Variable Slots:	(name (*PREP-PHRASE* 1))
Constraints:	(class JV-ENTITY *PREP-PHRASE*)
Constant Slots:	(type jv-entity subtype company relationship jv-partner)
Enabling Conditions:	(noun-triggered)

Figure 3.16: Concept node for "fins with <entity>"

3.5.2.4 Changes to the AutoSlog Interface

For MUC-5, we added a new feature to the AutoSlog interface that automatically proposes morphological variations of the original concept node definitions. We call it the *generalization module* and refer to the concept nodes created by the module as "generalized concept nodes".[32]

[31] Although this depends on the size of the corpus. If the training corpus is very small then even the most common words for the domain may appear only a few times in the corpus.

[32] This module was partly motivated by the fact that UMass/MUC-4 system contained a morphological analyzer but the UMass/MUC-5 system did not. The concept nodes created by AutoSlog for MUC-4 were attached to root words and CIRCUS automatically applied them to all morphological variations of the root words. Since the MUC-5 system did not do morphological analysis, AutoSlog created a separate concept node for each morphological variant that occurred during training. One could argue that morphological variants that do not occur in the training

The new concept nodes are not necessarily more general than the original ones, but they are created by generalizing from the originals. The list of possible generalizations is shown below:

1. singular noun → plural noun
2. plural noun → singular noun
3. passive verb → active verb
4. active verb → passive verb, additional active verb tenses
5. verb direct-obj → verb alone, direct object alone

When a user accepts a concept node definition, the generalization module tries to create morphological variations of the original. For example, if the user accepts the definition for the pattern "venture with <entity>" then the interface dynamically creates a definition for the plural form,"ventures". The user is then shown the definition for "ventures with <entity>" and asked whether the new definition should also be added to the dictionary. If so, then both definitions are added to the dictionary; if not, then only the original is saved. None of the generalized concept nodes are added to the dictionary without confirmation from the user!

The verb generalizations are slightly more complicated. Passive verb forms are converted to active verb forms, for example "<entity> was formed" is converted to "formed <entity>". And passive verbs followed by a prepositional phrase with "by" (e.g., "was formed by <entity>") are converted to the active form (e.g., "<entity> formed"). Given an active verb form, the interface also automatically generates definitions for four different verb tenses: present singular, present plural, simple past, present participle. For example, given the pattern "<entity> formed" (simple past), the interface automatically generates definitions for the variations "<entity> forms" (present singular), "<entity> form" (present plural), and "<entity> forming" (present participle). Simple morphology routines are used to generate the appropriate verb forms.[33]

In addition, two special generalization routines are used with the definitions created by the **<subject> verb direct-object** heuristic. Before the user begins filtering the dictionary, AutoSlog collects all of the **<subject> verb direct-object** definitions that share the same verb or the same noun. For example, if AutoSlog created definitions for the patterns "<entity> formed venture" and "<entity> formed company" then these definitions are put into one pile, and if AutoSlog created definitions for the patterns "<entity> formed venture" and "<entity> joined venture" then these are put into another pile. If AutoSlog created more than one[34] **<subject> verb direct-object** definition with the same verb then the interface generates a new definition that drops the noun, e.g. "<entity> formed". The rationale behind this rule is that, if AutoSlog proposed multiple patterns for the same verb with different direct objects, then the verb itself is often enough. The new pattern is more general than the old ones because it is activated by the verb regardless of what the direct object is. Again, the user is given the option of keeping the new definition or rejecting it.

---

corpus may be missing for a reason; that is, the variants have a different meaning that is not relevant to the domain. This is an interesting question, although the answer depends on the size of the training corpus as well.

[33]The morphology routines do not always generate the correct verb forms so the user can correct them if necessary.

[34]A constant can be set to control how many different instances need to be seen before the generalization is applied. We set the constant to 2 for these experiments.

Similarly, if AutoSlog created more than one <**subject**> **verb direct-object** definition with the same noun then the interface generates a new definition that drops the verb, e.g, "<entity> <verb> venture". This pattern is triggered by all occurrences of the noun "venture" that are preceded by a verb and extracts the subject of the verb as an entity. Concept nodes produced by this generalization routine are risky because they can be activated by many different expressions. They are useful, however, when the direct object carries the relevant semantic information and the preceding verbs are typically weak. For example, in the joint ventures domain AutoSlog proposed many different concept nodes for various expressions involving the noun "venture", such as "<entity> formed venture", "<entity> join venture", "<entity> started venture", "<entity> studying venture", "<entity> discussing venture", etc. In the MUC-5 corpus, almost any verb appearing before the noun "venture" produces a relevant expression for the joint venture domain; even negative expressions such as "IBM canceled the venture" are indicators of joint venture activity. The more general pattern may occasionally extract information from misleading expressions, such as "Honda hoped the venture between Nissan and Toyota would not be successful" but, in this case, the human-in-the-loop decided that the additional coverage provided by the generalized pattern was probably worth the risks associated with it.

3.5.2.5 Results for JV

We evaluated AutoSlog in the joint ventures domain using the MUC-5 corpus. The input to AutoSlog consisted of 10,684 string fills that came from 924 texts.[35] Table 3.6 shows the breakdown of string fills by slot. As Table 3.6 shows, some slots contained more fills than others. The majority of string fills came from the entity and product/service slots. As a result, most of the concept node definitions proposed by AutoSlog were for entities and product/service descriptions. Table 3.7 shows the breakdown by slot for the number of definitions proposed by AutoSlog, the number of definitions kept by the human-in-the-loop, and the percentage of definitions kept.

Table 3.6: Number of input strings by slot

Slot Name	**#String Fills**
entity name	3456
entity aliases	1233
facility name	97
ownership percent	814
ownership total capitalization	139
person name	554
product/service	4296
revenue rate	50
revenue total	45
TOTAL	10,684

[35]One of these texts was reclassified as irrelevant during the course of MUC-5. Therefore only 923 of these texts were considered to be relevant for the text classification experiments described in Section 4.6.3.

Table 3.7: Core AutoSlog dictionary for joint ventures

Slot Name	#CNs Proposed	#CNs Kept	%CNs Kept
entity name	1562	527	18%
facility name	80	20	34%
ownership percent	174	90	52%
ownership total capitalization	25	14	56%
person name	243	119	49%
product/service	1034	138	13%
revenue rate	19	14	74%
revenue total	30	22	73%
TOTAL	3167	944	30%

As in the terrorism domain, the percentage of definitions kept by the user varies across the different slots. In particular, the slots corresponding to monetary amounts (ownership total capitalization, revenue rate, revenue total) and percentages (ownership percent) have over a 50% retention rate. The expressions relating to these figures are often self-contained (e.g., "capitalized at <capitalization>") and these values can be easily mapped back to the source text. Many of the bad ownership percent definitions came from constructions such as "... formed a 50-50 joint venture", where the percentage is buried in a noun phrase. AutoSlog does not distinguish between cases where the targeted information is a noun modifier as opposed to the head noun. In general, when the targeted information is a noun modifier, AutoSlog proposes a pattern that is usually not useful. A possible modification to AutoSlog would be to inhibit AutoSlog from creating definitions when the targeted information does not include the head noun.

AutoSlog had the most difficulty with the product/service descriptions and entities. The product service descriptions were often long and included verbs and prepositional phrases (e.g., "materials used for civil engineering projects" or "providing a broad range of research investment consulting"). AutoSlog was designed to search for simple noun phrases and cannot currently handle input strings that include verbs. Although the most important noun in the product/service descriptions was usually marked (e.g., "materials" and "research" in the preceding examples), these nouns were often too general by themselves and AutoSlog either found the wrong reference to the noun or created a pattern that was overly general.

The low percentage associated with the entity slot was partially the result of entity aliases (e.g., "IBM") and governments ("Chinese"). In news articles, the full name of a company usually appears first (e.g., "International Business Machines Corp.") and an alias (e.g., IBM) is used for subsequent references. Therefore, the alias often appeared in sentences that did not explicitly mention the joint venture (violating the first AutoSlog assumption described in Section 3.2). Governments are often referred to implicitly using adjectives (e.g., "a Chinese-American joint venture") so AutoSlog often had trouble finding the references to governments in the original source text.

Table 3.8 shows the final statistics for the JV dictionary created by AutoSlog, including the generalized concept node definitions produced dynamically by the interface. The human-in-the-loop took 20 hours to review the 3167 concept nodes proposed by AutoSlog.[36] This is substantially

[36]The human-in-the-loop for the JV dictionary was the author, who was not the human-in-the-loop for the terrorism dictionary.

more time than it took the human-in-the-loop to review the terrorism definitions (5 hours).[37] The increased time is due to two factors. First, AutoSlog proposed 2.6 times as many definitions for the JV domain (3167) as for the terrorism domain (1237), primarily because AutoSlog received 2.2 times as many input strings for the JV domain (10,684) as for the terrorism domain (4780).

Second, a lot of the increased filtering time is due to the overhead associated with the new generalization module. The interface dynamically created one or more generalized definitions for the user to review each time the user accepted one of the originals. This substantially increased the number of definitions displayed to the user. In addition, some of the generalization routines required human interaction (e.g., the morphology routines often generated bogus verb forms that the user had to correct) which added many keystrokes to the filtering process. Consequently, the filtering processes for JV and terrorism were substantially different and therefore not comparable.

The statistics for the final JV dictionary are shown in Table 3.8. The first column shows the number of definitions originally proposed by AutoSlog. The second column shows the number of original definitions kept by the user. The third column shows the total number of definitions kept by the user, including the generalized definitions created by the interface. The generalized definitions increased the size of the dictionary by a factor of 2.7. This implies that, on average, each original definition spawned 1.5 generalized definitions.

Table 3.8: Generalized AutoSlog dictionary for joint ventures

Slot Name	**#CNs Proposed**	**#CNs Kept**	**#CNs Kept with Generalizations**
entity	1562	527	1570
facility name	80	20	38
ownership percent	174	90	184
ownership total capitalization	25	14	16
person name	243	119	355
product/service	1034	138	273
revenue rate	19	14	22
revenue total	30	22	57
TOTAL	3167	944	2515

AutoSlog created definitions for many expressions commonly associated with joint ventures and expressions that were not necessarily common but appropriate for the domain. The concept nodes proposed most frequently by AutoSlog (see Section 3.5.1) represented important expressions relating to joint venture activities. For example, AutoSlog proposed many definitions involving the noun "venture" (e.g., "venture with X", "venture between X", "X formed venture", "X set up venture"), and the word "agree" (e.g., "agreement with X", "X agreed", "X signed an agreement",

[37] It is possible that some of the time difference can be attributed to the different users. However, the human-in-the-loop for JV was probably at least as fast as the human-in-the-loop for terrorism because she had extensive experience with AutoSlog. Therefore, if anything, the disparity between the filtering time for JV and terrorism is likely even more pronounced.

"X agreed to form").[38] AutoSlog also identified many expressions that are somewhat idiosyncratic but relevant, such as "tie-up with X" or "X linked up".

Table 3.9: Frequently proposed patterns for JV

Linguistic Pattern	Number of Times Proposed
venture with <entity>	230
agreement with <entity>	54
venture between <entity>	51
<entity> formed venture	45
was owned by <entity>	39
<entity> agreed	38
<entity> set up venture	37
<entity> was capitalized	35
subsidiary of <entity>	34
<entity> signed agreement	34
unit of <entity>	34
PERCENT-OBJECT by <entity>	29
<entity> agreed to form	27

We did not have a hand-crafted dictionary with which to compare the AutoSlog dictionary for the joint ventures domain so it is difficult to assess the performance of the dictionary in a quantitative manner.[39] Therefore, we judged the quality of the dictionary by manually inspecting the definitions proposed by AutoSlog and by observing the performance of the system as a whole. During the course of MUC-5, we watched CIRCUS process many texts using the AutoSlog dictionary and felt satisfied that the dictionary extracted the majority of relevant information. Although the final scores for the UMass/MUC-5 system in the JV domain were not as high as the UMass/MUC-4 terrorism scores, we did not attribute the lower performance to poor dictionary

[38]Many of these expressions are often used with conjunctions, e.g., "venture between Toyota and Nissan" or "Toyota and Nissan agreed"; if a concept node finds a conjunction in the text then it extracts all of the conjuncts.

[39]In principle, one would like to determine how many correct and incorrect fillers were extracted by the concept nodes. However there are several factors that make this difficult to determine automatically. The main complication involves coreference. For example, the president of a company may be referred to multiple times in a text as "CEO Mr. Stanley Bingham", "Mr. Bingham", "the CEO", "the president", "he", etc. If a concept node extracts any of these references then it did the right thing. However, only one of the references, usually the most specific one, will appear in the key template. This makes it difficult to assess the false hit rate of the concept nodes automatically.

coverage.[40] Our final assessment of the UMass/MUC-5 system was that the AutoSlog dictionary provided us with excellent coverage for the joint ventures domain.

3.5.3 The Microelectronics Domain

We also used AutoSlog to create a concept node dictionary for the MUC-5 domain of microelectronics. We focused on 12 specific types of information found in the ME templates. Examples of legitimate fillers for each slot are shown in Table 3.10.

Table 3.10: Targeted information for microelectronics

Slot Name	Description	Example
bonding type	set fill	LASER_BONDING
device function	set fill	MICROPROCESSOR
device size	number & set fill unit	64 MBIT
device speed	number & set fill unit	70 MHZ
entity name	company, person, or govt.	"Material Research Corp."
equipment name	name or model number	"Precision 8000"
equipment type	set fill	CVD_SYSTEM
film type	set fill	SILICON_DIOXIDE
granularity size	number & set fill unit	LINE WIDTH 0.25MI
material type	set fill	CERAMIC
pin count	number	408
process type	set fill	CHEMICAL VAPOR DEPOSITION

As we discussed in Section 2.3.2, most of the information in the microelectronics templates is in the form of set fills, not strings. Only two of the twelve slot types (entity names and equipment names) contain string fills. AutoSlog expects string fills as input because they can be easily mapped back to the source text using a simple string search. Set fills are a problem because a set fill is a symbol that refers to a class name, but is not part of the original source text. That is, the set fills do not identify which parts of the text are relevant.[41]

To work around this problem, we exploited the MUC-5 microelectronics domain guidelines. The guidelines included a list of common terms and phrases that are relevant to each of the set fill classes. For example, the guidelines state that the set fill ACTIVE_DISCRETE_DEVICE applies to transistors, josephson junctions, quantum structures, bipolar devices, diodes, and opto-electronic devices. The guidelines were written to help the MUC-5 participants understand

[40]See [MUC-5 Proceedings, 1993] for a detailed assessment of the UMass/MUC-5 system. We believe that the weak link was the discourse analysis component which is responsible for mapping the extracted information into the final templates. Most, or even all, of the relevant information can be extracted from a sentence but the system will receive no credit for the information if the discourse analyzer does not put it into the correct template slot.

[41]Again, this is only a problem because we used the key templates as input to AutoSlog. An annotated corpus, such as the one proposed in Section 5.4.3, would contain markings to indicate which pieces of information are relevant.

the microelectronics domain and how to fill templates. For AutoSlog's purposes, we used these lists to develop special procedures that automatically map the set fills back to the original source text. For each set fill found in a template, AutoSlog searched the source text for any of the terms associated with the set fill in the guidelines. For example, given ACTIVE_DISCRETE_DEVICE as input AutoSlog searched the source text for the first reference to transistors, josephson junctions, quantum structures, bipolar devices, diodes, or opto-electronic devices. We also augmented the original lists to include singular and plural forms of the same term and different hyphenated variations.

The microelectronics domain is fundamentally different from the terrorism and joint venture domains because it is a technical domain. Terrorism and joint ventures revolve around actions (e.g., bombings in terrorism and forming companies in JV) so they are best characterized as event-driven domains. In contrast, the microelectronics domain revolves around objects and processes. Although there are some events in the ME world (e.g., a company develops an ME technique), most of the information that needs to be extracted is technical in nature and does not depend on events. The distinction between event-driven domains and technical domains is discussed in more detail in Chapter 5.

3.5.3.1 Sample Concept Node Definitions for ME

We created a dictionary for the microelectronics domain using the exact same version of AutoSlog that we used for the joint ventures domain. Because of the technical nature of the ME domain, AutoSlog did not generate many useful patterns for some types of information. We will illustrate the issues associated with technical domains by looking at some examples of concept nodes generated for microelectronics.

AutoSlog was most successful at generating concept node definitions to extract entities. The reason is that entities are where the action is. According to the MUC-5 domain guidelines, a relevant entity must be a company, government, or person that plays the role of developer, manufacturer, distributor, purchaser, or user. AutoSlog was designed specifically to extract role objects, i.e. objects that play a specific role in an event. Microelectronics entities must be involved in specific types of events (developing, manufacturing, distributing, purchasing, or using) to be relevant.

As an example, Figure 3.17 shows a concept node produced by AutoSlog to extract entities. This concept node is activated by the pattern "<X> developed technology" and extracts X. This pattern is a useful because it represents one of the five relevant actions associated with entities (i.e., developing).

Figure 3.18 shows another good concept node proposed by AutoSlog for the microelectronics domain which recognizes the pattern "researchers at <X>". Although this pattern does not explicitly refer to one of the relevant actions associated with entities, it is useful because the pattern is likely to extract the names of companies and organizations that are relevant. Of course, some of the entities extracted by this pattern will not be involved in microelectronics activities. But many of them will. As we explained in Section 3.5.1, concept nodes are useful if they are likely to extract a lot of relevant information for the domain even if they also extract irrelevant information sometimes.

Figure 3.19 shows a concept node produced by AutoSlog to extract microelectronics processes, such as layering, lithography, etching, or packaging. The relevant process in the text is MBE, which stands for "molecular beam epitaxy". AutoSlog identified "MBE" as the direct object of the verb "using" and created a concept node for the pattern "using <X>" to extract process types. Although the pattern works in this particular text, it is too general to reliably pick up microelectronics processes in future texts. The verb "using" is vague and can appear in many different contexts.

Id: 2533698 **Slot filler:** "Fujitsu Laboratories"
Sentence: Fujitsu Laboratories has developed a technology to selectively form a two-dimensional electron gas layer on top of an electron donor layer.

CONCEPT NODE

Name:	me-entity-subject-verb-and-dobj-developed-technology
Trigger:	technology
Variable Slots:	(name (*SUBJECT* 1))
Constraints:	(class ME-ENTITY *SUBJECT*)
Constant Slots:	(type me-entity subtype company relationship (developer))
Enabling Conditions:	(dobj-preceded-by-verb 'developed 'technology)

Figure 3.17: Concept node for "<entity> developed technology"

Id: 2527636 **Slot filler:** "Tokyo Institute Of Technology's"
Sentence: Researchers at the Tokyo Institute of Technology's Research Laboratory for Engineering Materials have reportedly developed a process to form a bi-based superconducting thin film which involves processing temperatures of 300 to 380 deg C, and uses an MBE (molecular beam epitaxy) method.

CONCEPT NODE

Name:	me-entity-pp-noun-researchers-at
Trigger:	researchers
Variable Slots:	(name (*PREP-PHRASE* (pp-check '(at))))
Constraints:	(class ME-ENTITY *SUBJECT*)
Constant Slots:	(type me-entity subtype company relationship (developer))
Enabling Conditions:	(noun-triggered)

Figure 3.18: Concept node for "researchers at <entity>"

Id: 2533698 **Slot filler:** "MBE"
Sentence: To form the layer, the laboratory developed a continuous process for growing crystals in an ultra-high vacuum environment using MBE, a method of selectively implanting impurities with an FIB (focused ion beam) method, and adopted a high-speed heat treating process.

CONCEPT NODE

Name:	me-process-type-dobj-verb-using
Trigger:	using
Variable Slots:	(name (*DOBJ* 1))
Constraints:	(class ME-PROCESS *DOBJ*)
Constant Slots:	(type me-process subtype layering)
Enabling Conditions:	(active)

Figure 3.19: Concept node for "using <process>"

This example illustrates why technical domains are not well-suited for AutoSlog. Technical information is often wholly contained in noun phrases and does not rely on surrounding linguistic cues. For example, molecular beam epitaxy (MBE) refers to a specific microelectronics process; its meaning is unambiguous and will not change in different contexts. For the purposes of information extraction, technical jargon can usually be reliably extracted using simple keyword matching. In most cases, the surrounding context does not provide any additional help in identifying the relevant information.

However, sometimes it is important to identify the events associated with technical information. For example, in the MUC-5 domain, microelectronics processes are relevant only if they are being developed, manufactured, distributed, purchased, or used by an entity. Keyword matching alone would not be able to distinguish between texts that just mention a microelectronics process from texts that describe how a microelectronics process is being used or developed by a specific company. In short, if the information extraction task only involves finding technical information then keyword matching is appropriate. However, if the information extraction task also involves identifying roles associated with the technical information then additional linguistic context must be used. In Chapter 5, we characterize domains and tasks in more detail and explain which types are appropriate for AutoSlog.

Since microelectronics is a technical domain, the AutoSlog definitions were not very useful for identifying technical concepts. However, because the relationship between microelectronics processes and entities is important for the MUC-5 domain, keyword recognition by itself is not sufficient to identify the entities and the roles that they play. Therefore we adopted a two-stage approach that combined keyword recognition with the AutoSlog dictionary. First, we used AutoSlog to generate a concept node dictionary for the microelectronics domain just as we did for terrorism and joint ventures. However, we added a filter to the information extraction system to ensure that each instantiated concept node extracted a phrase related to microelectronics. For example, the AutoSlog dictionary contains a concept node for the pattern "using <X>". This concept node is activated by all occurrences of the word "using", but the filter throws away all instantiations that do not extract a microelectronics term such "molecular beam epitaxy" or "chemical vapor deposition". The keywords act as constraints to prevent the system from extracting irrelevant information, while the concept nodes allow the system to identify the role of the microelectronics process (e.g., whether it was used, developed, etc.).

This two-phase approach involves a tradeoff with respect to portability. On one hand, the system requires a predefined set of keywords for the domain; for the microelectronics domain, we provided the system with a set of terms that refer to relevant microelectronics processes and devices. On the other hand, this approach potentially reduces the amount of time required for a person to manually filter the dictionary. The primary reason that the AutoSlog dictionary needs to be filtered is to remove concept nodes that are likely to extract mainly irrelevant information. The keyword filtering process, however, guarantees that the system will only output relevant information. There are other reasons why the AutoSlog dictionary may need to be filtered (e.g., to eliminate concept nodes that represent irrelevant event types) but they typically represent only a small fraction of the dictionary.

For the microelectronics domain, we applied the two-phase keyword filtering to the concept nodes for 10 of the 12 slots (the setfill slots) and did not manually filter these definitions at all. However, for the 2 slots that can be filled with arbitrary strings (the entity name and equipment name slots), we put people in the loop to manually filter the concept nodes just as we did for the terrorism and joint ventures domains.

3.5.3.2 Results for ME

We applied AutoSlog to 787 relevant texts from the MUC-5 development corpus.[42] We targeted the 12 slots types outlined in Section 3.5.3, which consist of 2 slots that accept string fills (entity names and equipment names) and 10 slots that accept set fills. As we just explained, the concept nodes produced by AutoSlog for the set fill slots were too general so we used keyword filtering as a postprocessing step in the information extraction system. Since the keywords extracted by the concept nodes uniquely determine which template slot should be filled, there was no need to distinguish the concept nodes by slot type. Therefore we merged all of the concept nodes generated for the 10 set fill slots and renamed them all as *setfill types*. In practice, merging them has two side effects: (1) it reduces the size of the dictionary because many of the different set fill types produced identical concept node patterns (e.g., "using <X>") and (2) it increases the coverage of the dictionary because concept nodes generated by one slot are now capable of extracting information for another slot.

Table 3.11 shows the statistics for the microelectronics dictionary. AutoSlog proposed 2952 concept node definitions for the microelectronics domain; 2275 of these definitions were saved in the final dictionary. As we just mentioned, the entity name and equipment name concept nodes were manually filtered[43] because they extract arbitrary strings, but the setfill concepts nodes were not filtered by a human.[44] The total filtering time for the ME dictionary was 15.5 hours, which consisted of 10.5 hours to manually filter the entity name and equipment name concept nodes plus 5 hours to "automatically" filter the setfill concept nodes. Even though we kept all of

[42]One of these texts was classified as relevant when we did these experiments but was reclassified as irrelevant by the MUC-5 organizers before the final evaluation. Therefore the MUC-5 microelectronics corpus officially contains only 786 relevant texts.

[43]Two people split up the task of filtering the microelectronics dictionary: one was a second-year graduate student who had some but not extensive experience with AutoSlog and the other was a third-year graduate student who had previously filtered the terrorism dictionary (but he was a second-year graduate student at that time). The third-year student did most of the filtering because he was more familiar with the microelectronics domain.

[44]Table 3.11 shows that only 1728 of the 1732 setfill definitions were actually kept. This is because one of the users inadvertently threw away four of the bad setfill definitions.

the setfill definitions, we needed the generalization module in the AutoSlog interface to generate morphological variations so someone actually needed to accept each definition manually.[45]

Table 3.11: Core AutoSlog dictionary for microelectronics

Slot Name	#Proposed	#Kept	%Kept
entity name	971	451	46%
equipment name	249	96	39%
setfill type	1732	1728	100%
TOTAL	2952	2275	77%

We also applied the generalization module in the AutoSlog interface to the microelectronics dictionary. Table 3.12 shows the size of the dictionary when the generalized concept nodes are included. Notice that we included the generalized definitions for the set fill concept nodes even though they weren't manually filtered by a user. The generalized concept nodes substantially increase the size of the dictionary. The final microelectronics dictionary contained 4220 concept node definitions.

Table 3.12: Generalized AutoSlog dictionary for microelectronics

Slot Name	#Proposed	#Kept	#Kept with Generalizations
entity name	971	451	1445
equipment name	249	96	209
setfill type	1732	1728	2566
TOTAL	2952	2275	4220

As with the joint ventures domain, there was no hand-crafted dictionary with which to compare the AutoSlog dictionary for the microelectronics domain. In general, however, we felt that the AutoSlog dictionary provided adequate coverage for the entity and equipment slots. Table 3.13 shows the ten most frequently proposed concept nodes for these slots. The patterns are not as specific as those for the joint ventures domain, but most of them are likely to extract relevant information for the ME domain.

However, most of the domain-specific terms for microelectronics came from the keywords that were used to filter the setfill concept nodes. In theory, the concept node patterns are necessary in order to understand the relationship between the microelectronics objects and the entities (e.g., whether a particular device was developed or used by a company). It is not clear how much the UMass/MUC-5 system actually benefited from the concept nodes themselves. It is possible that the keywords alone would have worked nearly as well. The answer to this question rests with the discourse analyzer, which is the component of the UMass/MUC-5 system that used the concept

[45]Theoretically, this could be fully automated but we were operating under strict time constraints and did not have time to automate this process.

Table 3.13: Frequently proposed patterns for microelectronics

Linguistic Pattern	Number of Times Proposed
agreement with <entity>	18
researchers at <entity>	17
order from <entity>	14
manager at <entity>	14
includes <equipment-name>	13
<entity> developed technology	12
was developed by <entity>	12
order for <equipment-name>	11
introduced <equipment-name>	11
include <entity>	10

nodes. However, the discourse analyzer was not developed at UMass[46] and it would be difficult for us to delve into it to analyze the effect of the dictionary on its performance. Ideally, one would like to evaluate the system without the setfill concept nodes at all but this would require developing a new discourse module.

In summary, we believe that the AutoSlog dictionary provided good coverage for the string fill slots in microelectronics. On the other hand, technical information such as the names of specific microelectronics devices and processes are best handled by keywords. However, AutoSlog can generate concept nodes that may be useful for determining the roles that these objects play in events.

3.6 Experiments with Novice Users

In order for information extraction systems to be portable across domains, tools for automated knowledge acquisition and rapid prototyping are essential. However, it is important to remember that the ultimate users of these tools will be domain experts, not natural language processing researchers. Domain experts have extensive knowledge about the task, but have little or no background in linguistics or text processing. Tools that are accessible only to fellow researchers will be of limited use in real-world scenarios.

With this in mind, we conducted two experiments to see whether people with little or no background in natural language processing could use AutoSlog effectively. Our goals were to find out whether:

1. Anyone with knowledge of the domain can use AutoSlog to create a concept node dictionary, with minimal training.
2. Dictionaries created by novices can achieve good performance.

Furthermore, we were interested to see how well dictionaries created by different people would perform, and how consistent they would be with each other. The first experiment involved ten students with some background in natural language processing, who used AutoSlog to create dictionaries for the terrorism domain. The second experiment involved two government analysts who had no background in natural language processing but were domain experts for the joint ventures domain.

[46]The discourse analyzer (TTG) was developed by our collaborators at Hughes Research Labs (see [Lehnert *et al.*, 1993a, Lehnert *et al.*, 1993b]).

3.6.1 An Experiment with Students

The first experiment with novice users involved ten students in the introductory natural language processing course at the University of Massachusetts. The class consisted of both undergraduates and graduate students. During the course, the students received some exposure to CIRCUS, including 2 lectures, 1 paper to read, and 2 programming assignments. They also had some exposure to the information extraction task for terrorism, including 1 lecture and 1 paper. So the students had some background in natural language processing and CIRCUS in general, but no experience with the UMass/MUC-4 system (with the exception of one student in the class who was also in our lab; we will refer to him as Student X).

Before the experiment began, the students were given 1 hour of instruction explaining how to use the AutoSlog interface. They were given two weeks to build their own dictionaries for terrorism using the interface. The experiment was a requirement for the course but no grades were given so the students were not evaluated on the quality of their dictionaries.

Once the experiment was completed, we compared the performance of the students' dictionaries with the performance of the hand-crafted MUC-4 dictionary. For each student dictionary, we took the official UMass/MUC-4 system, replaced the hand-crafted dictionary with the student dictionary, and scored the resulting system using the MUC-4 scoring program. We scored each system on both TST3 and TST4. The results are shown in Table 3.14.

Two of these data points are somewhat anomalous. Student X was a member of our lab so he had some knowledge about the UMass/MUC-4 system, but he did not have extensive experience with the system. Nevertheless, his results should not be interpreted as those of a novice.[47] The second anomalous data point is Student I. Student I was not a native speaker of English and he apparently did not understand the instructions. When the experiment was over, we found that he had kept every concept node proposed by AutoSlog; he did not throw any of them away! As a result, the scores generated by Student I's dictionary represent a baseline; his scores reflect the performance of the system with an unfiltered AutoSlog dictionary.

If we disregard the data points associated with Student X and Student I, we see fairly consistent results across the different dictionaries. For TST3, the scores range from 70-87% of the performance of the hand-crafted MUC-4 dictionary (based on the F-measure). For TST4, the scores range from 67-94% of those for the MUC-4 dictionary. Although there is quite a range of performance, the majority of the dictionaries achieve about 75-85% of the performance of the MUC-4 dictionary. Figure 3.20 shows the scatterplots for the recall and precision scores.

To put these numbers in perspective, consider how the scores of the student dictionaries compare with the scores of the MUC-4 participants. Once again, we will disregard the scores for the dictionaries produced by Student X and Student I. The best of the student dictionaries achieved an F-measure of 43.82 on TST3, which would have placed it fifth in the MUC-4 rankings. That is, only four of the seventeen MUC-4 systems achieved higher scores on TST3. At the other end of the spectrum, the student dictionary that obtained the lowest score of 35.57 would have ranked eighth in MUC-4. So all of the student dictionaries achieved scores better than half of the MUC-4 participants on TST3. On TST4, the student dictionary with the highest score would have ranked seventh and the dictionary with the lowest score would have ranked eleventh. In general, we conclude that the concept node dictionaries generated by students achieved scores that were comparable or better than many of the other MUC-4 systems.

One explanation for the relatively consistent results across different dictionaries is that some definitions are more important than others; a subset of the definitions are used more frequently than the rest. In other words, there is probably something like an 80/20 rule in effect where 20%

---

[47]The AutoSlog results presented in Section 3.5.1 were based on Student X's dictionary. We used his dictionary as a basis for comparison against the hand-crafted dictionary because the hand-crafted dictionary was created by an experienced system developer.

Table 3.14: Student dictionary scores on TST3 and TST4

TST3

System	Recall	Precision	F-measure
MUC-4	46	56	50.51
Student X	43	56	48.65
Student A	39	50	43.82
Student B	38	44	40.78
Student C	33	52	40.38
Student D	38	43	40.35
Student E	36	42	38.77
Student F	37	38	37.49
Student G	34	39	36.33
Student H	31	42	35.57
Student I	31	17	21.96

TST4

System	Recall	Precision	F-measure
MUC-4	44	40	41.90
Student X	39	45	41.79
Student A	37	42	39.34
Student C	30	41	34.65
Student D	35	34	34.49
Student H	31	38	34.14
Student B	33	34	33.49
Student E	31	36	33.31
Student G	32	32	32.00
Student F	28	28	28.00
Student I	35	15	21.00

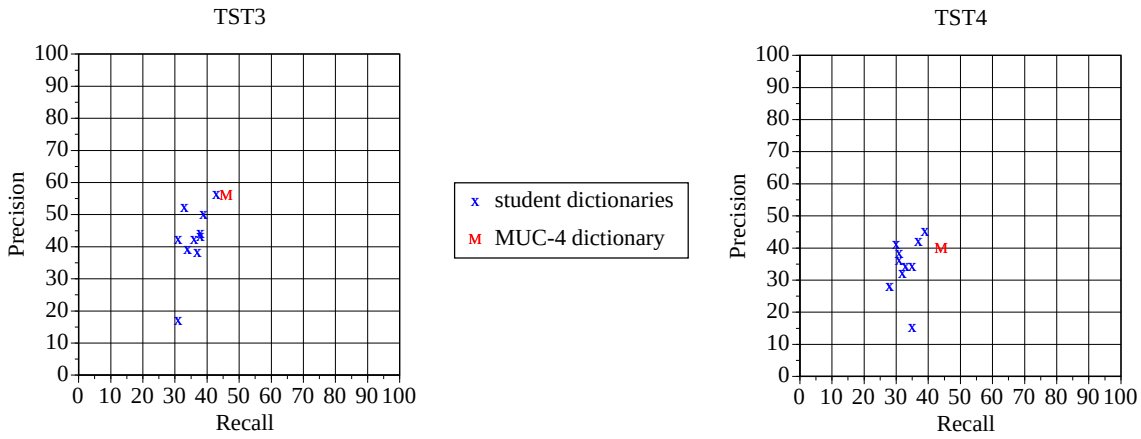


Figure 3.20: Recall and precision scores for the student dictionaries

of the concept nodes are doing 80% of the work.[48] Consequently, as long as a user retains the most important definitions, their dictionary will probably achieve relatively good performance. Notice that even Student I, who retained *every* definition, achieved recall levels that were comparable to the others.

We also tried to determine whether there were any correlations between dictionary size and performance. In general, one might assume that larger dictionaries will produce high recall and

Table 3.15: Student dictionary sizes

Dictionary	# of Definitions
Student C	304
MUC-4	389
Student A	390
Student H	399
Student B	422
Student X	450
Student E	478
Student D	567
Student G	619
Student F	645
Student I	1237

low precision but smaller dictionaries will produce low recall and high precision. But this is not necessarily the case. Figure 3.21 shows that the relationship between recall and dictionary size

[48]For the MUC-4 dictionary, we found that 18% of the definitions accounted for 80% of the instantiated concept nodes when processing the MUC-4 corpus (all 1700 texts). 28% of the definitions accounted for 90% of the instantiated concept nodes. Of the 365 definitions that fired at least once, nearly half (48%) fired $\leq$ 10 times.

and does not reveal any consistent patterns. Some of the smallest dictionaries produce the highest recall and some of the biggest dictionaries produce the lowest recall.

To understand this, one must remember that discourse analysis plays a major role in the UMass/MUC-4 system. The discourse analyzer handled problems such as co-reference resolution (i.e., determining when two items refer to the same object) and event recognition (i.e., determining the number of distinct events and mapping each object to the appropriate event). Discourse analysis becomes more complicated as the number of extracted items increases and becomes increasingly difficult when spurious information is introduced. When a concept node extracts spurious information, the discourse module may create spurious events to account for the information. The mapping procedures may also become confused and relevant pieces of information may be mapped to the wrong event. Consequently, even though a piece of information was correctly extracted, the MUC-4 scoring program will not give the system credit for it because the information was placed in the wrong template.

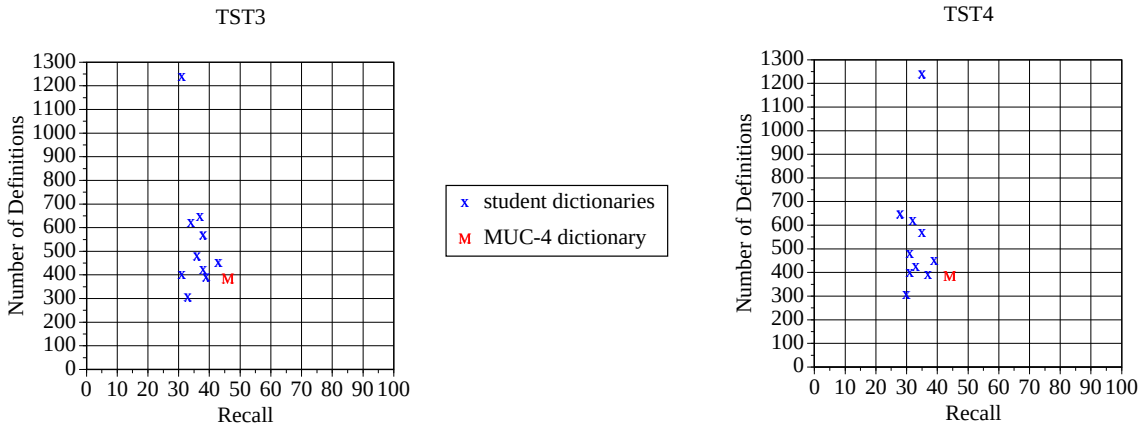


Figure 3.21: Recall vs. number of definitions

Figure 3.22 shows the relationship between precision and dictionary size. Although there is not a perfect correlation, the graph suggests that smaller dictionaries produce higher precision than bigger dictionaries. This makes sense if we assume that the smaller dictionaries represent conservative filtering strategies. Students using a conservative strategy probably retained the *most* reliable concept node definitions and rejected all definitions that might be prone to false hits. Therefore the smaller dictionaries contain concept nodes that are less likely to extract spurious information.

We also scored the student dictionaries on a different task called "text filtering". In MUC-4, text filtering refers to the problem of distinguishing the relevant texts (i.e., the ones that contain a relevant event description) from the irrelevant texts (i.e., the ones that do not contain a relevant event description). This is identical to the binary text classification task described in Chapter 4. The MUC-4 scoring program computes recall and precision scores for the text filtering task as well as the information extraction task. Figure 3.23 shows the text filtering scores for the dictionaries.

The results in Figure 3.23 are interesting for several reasons. First, there is less variation in scores across the different dictionaries. On TST3, all of the dictionaries achieved at least 80% recall and 78% precision. Many of them achieved $\geq$ 85% recall and $\geq$ 90% precision. On TST4, we see a few anomalous data points but most of the dictionaries achieve $\geq$ 84% recall and $\geq$ 74% precision. Second, the text filtering scores for the student dictionaries are very close to the text filtering scores for the hand-crafted dictionary, which obtained 91% recall and 94% precision on

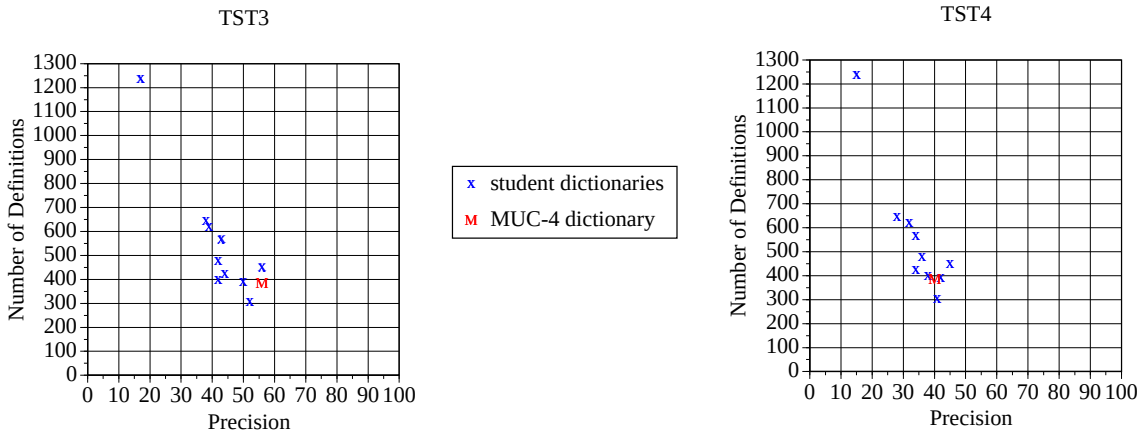


Figure 3.22: Precision vs. number of definitions

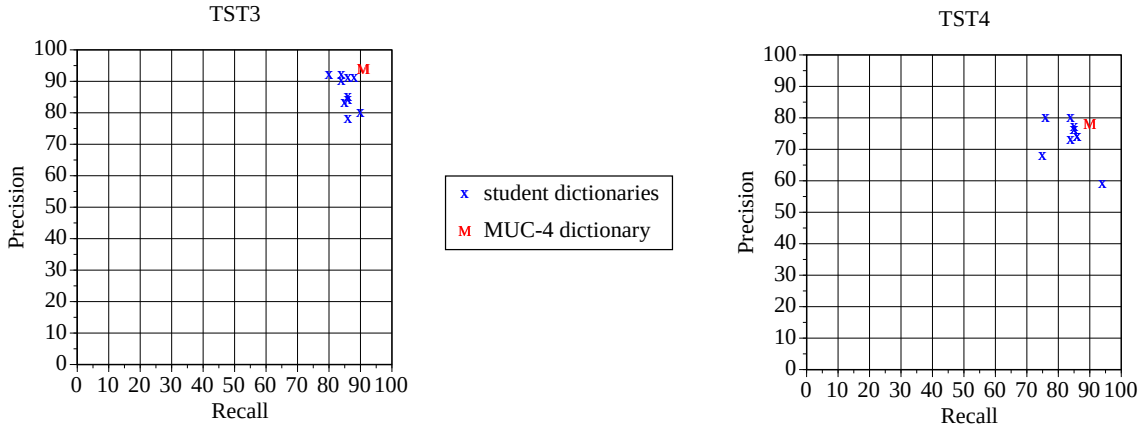


Figure 3.23: Recall and precision scores for text filtering

TST3 and 90% recall with 78% precision on TST4. If we view these scores as an upper bound, then the results for the student dictionaries are impressive because they did nearly as well.

To summarize, this experiment demonstrated that students with only minimal background in natural language processing can use AutoSlog effectively. Many of the dictionaries created by the students achieved performance levels only slightly lower than that of a hand-crafted dictionary. In general, conservative filtering strategies tend to produce dictionaries with relatively high precision, but sometimes at the expense of recall. Finally, dictionaries produced by different people achieved text filtering scores that were comparable to those for the hand-crafted dictionary and were generally consistent across different users.

3.6.2 An Experiment with Domain Experts

The goals of this experiment were:

1. to determine whether people with no background in text processing could use AutoSlog effectively.
2. to see what levels of performance could be obtained from dictionaries created by domain experts.

Two government analysts agreed to be the subjects of the experiment. The analysts were experts with the the JV domain and the template-filling task. Neither analyst had any background in linguistics or text processing or had any previous experience with the UMass/MUC-5 system. Before they began, we gave them a 1.5 hour tutorial to explain how AutoSlog works and how to use the interface. The tutorial included general decision-making advice and some examples to highlight important issues.

We did not give the analysts all of the concept node definitions proposed by AutoSlog for the JV domain. AutoSlog proposed 3167 concept node definitions, but the analysts were only available for two days and we did not expect them to be able to review 3167 definitions in this limited time frame. So we created an "abridged" version of the dictionary by eliminating entity and product/service definitions that were proposed infrequently by AutoSlog.[49] The resulting "abridged" dictionary contained 1575 concept node definitions.

We compared the analysts' dictionaries with the MUC-5 dictionary generated by a UMass researcher using AutoSlog. However, the UMass dictionary was based on the complete set of 3167 definitions originally proposed by AutoSlog as well as definitions that were spawned by AutoSlog's generalization module. We did not use the generalization modules in this experiment because of time constraints. To create a comparable UMass dictionary, we started with the MUC-5 dictionary that had been manually filtered by a UMass researcher and removed all of the "generalized" definitions as well as the definitions that were not among the 1575 given to the analysts. In other words, any concept node that was not given to the analysts was removed from the UMass dictionary. This process produced a much smaller subset of the UMass dictionary that was based on exactly the same set of inputs as the analysts' dictionaries but was filtered by a UMass researcher. Analyst A took approximately 12.0 hours and Analyst B took approximately 10.6 hours to filter their respective dictionaries. Table 3.16 shows the number of definitions that each analyst kept. For comparison's sake, we also show the breakdown for the abridged UMass dictionary.

---

[49]This was based on the frequency counts described in Section 3.5.1. We removed all entity definitions that were proposed < 2 times and all product/service definitions that were proposed < 3 times. We chose to eliminate entity and product/service definitions only because the sheer number of these definitions overwhelmed the others.

[50]The abridged version.

Table 3.16: Comparative dictionary sizes

CN Type	#proposed by AutoSlog	#kept (UMass[50])	#kept (Analyst A)	#kept (Analyst B)
entity	688	311	357	423
facility	80	20	16	55
ownership percent	174	91	117	91
person	243	119	149	52
product/service	316	76	152	44
revenue rate	19	14	12	16
revenue total	30	22	15	26
total capitalization	25	14	13	22
TOTAL	1575	667	831	729

To compare the dictionaries, we took the UMass/MUC-5 system, removed the official UMass dictionary, and replaced it with a new dictionary (an analysts dictionary or the abridged UMass dictionary).[51] Finally, we scored each new version of the UMass/MUC-5 system on the Tips3 test set that was used for the MUC-5 evaluation. Table 3.17 shows the results for each dictionary.

Table 3.17: Comparative scores for Tips3

TIPS3	Recall	Precision	F-measure	ERR
UMass/Hughes	18	51	27.06	83
Analyst A	19	47	27.39	83
Analyst B	20	47	27.89	83

The F-measures were extremely close across all 3 dictionaries. In fact, both analysts' dictionaries achieved slightly higher F-measures than the UMass dictionary. The error rates (ERR) for all three dictionaries were identical[52], but there is some variation in the recall and precision scores. We see more variation when we score the three parts of Tips3 separately[53] (see Table 3.18).

Overall, the analysts dictionaries achieved slightly higher recall but lower precision than the UMass dictionary. We hypothesize that this is because the UMass researcher was not very familiar with the corpus and was therefore somewhat conservative about keeping definitions. The analysts

[51]One complication is that the UMass/MUC-5 system includes two modules, TTG and Maytag, that use the concept node dictionary for training. Ideally, we should retrain these components using the new dictionary. We did retrain the template generator (TTG), but we did not have time to retrain Maytag. We expect that this should not have a significant impact on the relative performances of the dictionaries, but we are not certain of its exact impact.

[52]See [MUC-5 Proceedings, 1993] for a description of the error rate measure.

[53]The Tips3 test collection consisted of three separate sets of texts.

Table 3.18: Comparative scores for Part1, Part2, and Part3

TIPS3/Part1	Recall	Precision	P&R	ERR
UMass/Hughes	18	51	27.04	83
Analyst A	20	48	28.00	82
Analyst B	22	47	29.69	81

TIPS3/Part2	Recall	Precision	P&R	ERR
UMass/Hughes	17	52	26.03	84
Analyst A	18	48	25.92	84
Analyst B	20	47	27.75	83

TIPS3/Part3	Recall	Precision	P&R	ERR
UMass/Hughes	20	50	28.12	82
Analyst A	20	46	27.96	82
Analyst B	17	48	25.25	84

were much more familiar with the corpus and were probably more willing to keep definitions for patterns that were familiar to them. In general, there is a trade-off involved in making these decisions: a liberal filtering strategy often results in higher recall but lower precision whereas a conservative strategy results in lower recall but higher precision.

It is interesting to note that even though there was a lot of variation in the composition of the dictionaries (see Table 3.16), the resulting scores were very similar. As we explained in the previous section, we are probably seeing an 80/20 rule in effect where a core subset of the definitions, shared by all of the dictionaries, were exercised more than the others. This has important implications for system development: different dictionaries may achieve similar levels of performance as long as they share the same core set of important definitions. Consequently, if we could identify these core definitions a priori then we could significantly reduce the time needed for the human in the loop.

3.7 Summary

In this chapter, we presented:

- A system called AutoSlog that automatically constructs domain-specific dictionaries for information extraction given an appropriate training corpus.
- Results in the terrorism domain which showed that a dictionary constructed by AutoSlog achieved 98% of the performance of a hand-crafted dictionary. Furthermore, the hand-crafted dictionary required approximately 1500 person-hours to build but the AutoSlog dictionary required only 5 person-hours of effort and a training corpus.
- Results in the joint ventures and microelectronics domain which suggested that dictionaries created by AutoSlog provided good coverage in these domains. However, AutoSlog is not particularly well-suited for extracting technical information.
- An experiment with students which demonstrated that people with only minimal background in text processing can use AutoSlog effectively.

- An experiment which demonstrated that domain experts with no background in text processing can use AutoSlog effectively. The analysts' dictionaries produced scores that were better than a dictionary constructed by an NLP researcher, supporting the claim that AutoSlog is an effective tool for domain experts.

CHAPTER 4

INFORMATION EXTRACTION AS A BASIS FOR TEXT CLASSIFICATION

4.1 Text Classification

As storage capacities grow and memory becomes cheaper, we will soon have more information at our fingertips than we ever could have imagined. Already, we have databases that contain hundreds of thousands of documents. Intelligent information retrieval is essential to cope with such vast amounts of text. A central problem in information retrieval is *text classification* (or text categorization). The task is to automatically classify texts into one or more pre-defined categories. There are many applications that can take advantage of the classifications. For example, *text routing* applications automatically route texts to users who have a specific information need. *Text filtering* applications filter out texts that are classified as irrelevant so that only the relevant documents are passed on to the user (e.g., [Foltz and Dumais, 1992, Liddy *et al.*, 1993]). Text routing, text filtering, and text classification are all closely related areas of information retrieval [Belkin and Croft, 1992].

Traditional approaches to information retrieval use keyword searches and statistical techniques to retrieve relevant documents (e.g., [Turtle and Croft, 1991, Salton, 1989]). Statistical techniques take advantage of large document collections to automatically identify words that are useful indexing terms. These techniques are popular because they can be fully automated and can sift through large volumes of documents with relative ease. In general, however, word-based techniques have several limitations:

Synonymy: Different words and phrases can express the same concept. For example, the words "make", "manufacture", and "produce" all refer to the concept of production.

Polysemy: Words can have multiple meanings. For example, the word "post" can refer to the name of a newspaper, a vertical support in carpentry, entering a transaction in accounting, or sending a message in computing.[1]

Phrases: Some words are good indexing terms only in specific phrases. For example, the phrase "passed away" means that someone died but the words "passed" and "away", used independently, are not associated with dying.

Local context: Some words and phrases are good indexing terms only in specific local contexts. For example, to retrieve texts about bank robberies, the word "robbery" alone is not enough; the object of the robbery must be a bank.

---

[1] This example is from [Mauldin, 1991].

Global context: Some documents do not contain any words or phrases that are good indexing terms. The relevance of a document may depend on the entire context of a sentence, paragraph, or even the whole text. For example, the sentence "an armed man took the money and fled" clearly refers to a robbery even though none of the words are good indexing terms individually.

Synonymy is a well-known limitation of word-based techniques that can make it difficult to find relevant documents. Some IR systems access a thesaurus or an on-line dictionary to alleviate this problem (e.g., [Crouch, 1988, Crouch and Yang, 1992, Mauldin, 1991]). Word-sense disambiguation techniques have been used to investigate the issue of polysemy (e.g., [Krovetz and Croft, 1989]). The last three points address issues of linguistic context. Some IR systems have tried automatic phrase indexing methods that use multiple words together as indexing terms, for example [Croft *et al.*, 1991, Dillon, 1983, Fagan, 1989]. But these approaches only approximate phrasal recognition and provide a weak sense of context. By using multiple words or phrases to index a document, IR systems capture some global context but typically do not represent the relationships between words beyond co-occurrence statistics.

As an alternative to traditional IR systems, there has been a lot of work recently on knowledge-based information retrieval systems (e.g., [Goodman, 1991, Hayes and Weinstein, 1991, Mauldin, 1991, Rau and Jacobs, 1991]). Knowledge-based IR systems rely on an explicit knowledge base, such as a rule base [Hayes and Weinstein, 1991], semantic network [Goodman, 1991], patterns [Rau and Jacobs, 1991], or case frames [Mauldin, 1991]. Many of these systems have achieved good success in limited domains. However, knowledge-based approaches typically require an extensive manual knowledge engineering effort to create the knowledge base. Manual knowledge engineering is a time-consuming and tedious process that may require several person-years of effort by experts who are highly experienced with the domain and the task. To achieve similar success in a new domain, the entire knowledge engineering process must be repeated.

Both traditional IR techniques and knowledge-based techniques have been applied to the problem of text classification (e.g., see [Maron, 1961, Borko and Bernick, 1963, Hoyle, 1973] for traditional IR approaches and [Goodman, 1991, Hayes and Weinstein, 1991, Rau and Jacobs, 1991] for knowledge-based approaches). Text classification is an information retrieval task in which one or more category labels is assigned to a document. This task assumes a pre-defined, long-term set of user interests (categories) which is different from the standard information retrieval task that assumes dynamically changing user needs (queries) [Belkin and Croft, 1992]. However, both tasks share many of the same problems because they are primarily concerned with identifying relevant documents from large collections of raw text.

Our approach to text classification is a departure from standard IR techniques in several ways. First, we use a natural language processing task called *information extraction* as a basis for text classification. Although in-depth natural language processing can be prohibitively expensive and brittle, information extraction is a more tractable and robust technology. Using natural language processing, we can overcome many of the limitations imposed by word-based techniques. In particular, we consistently achieve high precision because our approach is sensitive to context. Linguistic phrases and context surrounding the phrases are recognized easily and handled naturally. In addition, the system can classify texts that would be inaccessible to some word-based techniques because they do not contain any key words or phrases.

Second, our approach is knowledge-based because it relies on a domain-specific dictionary to drive the information extraction system. However, the text classification algorithms are domain-independent and the domain-specific dictionary can be acquired automatically, given an appropriate training corpus. Therefore the complete text classification system is fully trainable and can be easily scaled up or ported to new domains. By automating the construction of a knowledge-based text classification system, we have greatly reduced the knowledge engineering bottleneck typically required for such systems while benefiting from a knowledge-based approach.

Finally, the emphasis of this research is on *high precision* text classification. In many real-world applications, users are satisfied to receive a small number of relevant documents, as long as they can be reasonably confident that the documents have been classified accurately. The algorithms that we will describe allow the user to specify how conservative or liberal the algorithms should be about assigning categories. In general, there is usually a tradeoff between retrieving as many relevant texts as possible and retrieving relevant texts accurately. Liberal algorithms are more eager to classify texts as relevant but may misclassify many texts. Conservative algorithms are more reluctant to classify texts as relevant so they generally produce fewer false hits but may fail to recognize many relevant documents. Although our text classification algorithms can achieve a broad range of results, our approach is particularly well-suited for applications in which the accuracy of the classifications is more important than recognizing every relevant document.

In this chapter, we describe three algorithms that use information extraction as a basis for text classification: the relevancy signatures algorithm, the augmented relevancy signatures algorithm, and a case-based text classification algorithm. We will refer to this general approach as *IE-based text classification*, where IE-based is shorthand for information extraction based. We present empirical results for all three algorithms in three different domains: terrorism, joint ventures, and microelectronics.

4.2 Motivation

Our work on text classification was motivated by three observations about how humans classify documents:

1. Human readers usually find some texts difficult to classify because they fall into gray areas with respect to the domain specifications. On the other hand, many documents are straightforward to classify because they fall squarely within the domain guidelines. A human can quickly and easily pick out these texts.[2] Our goal is to simulate this human process of recognizing the texts that are most likely to be relevant. By focusing on the relatively straightforward texts instead of the borderline cases, we are willing to miss some relevant texts in exchange for good accuracy on the ones that we do classify as relevant.

2. A single relevant sentence is often enough to classify a text as relevant. In some cases, as soon as an important expression is identified, a text can be accurately classified. For example, in the domain of terrorism, the expression "was shot to death" is a strong indicator of relevance. If a text in the MUC-4 corpus contains this expression, then we can quickly and confidently classify the text as relevant.

3. Once a relevant sentence is identified, the remainder of the text can be ignored. As soon as we find a relevant sentence, the text should be classified as relevant regardless of what appears in the remainder of the text.[3]

---

[2]In an informal experiment, we asked two graduate students to scan 100 MUC-4 texts and pick out any texts that they could quickly and confidently identify as relevant. The first student took 15 minutes to go through all 100 texts and achieved 83% recall and 96% precision. The second student took 30 minutes and achieved 86% recall and 94% precision. In this small amount of time, the students could not possibly have *read* all of the documents. Their good results support our claim that, in this domain at least, many documents can be accurately classified by text skimming.

[3]Points 2 and 3 are not always true, especially when the domain description contains many exceptions. Our algorithms assume that these exceptional cases are relatively infrequent in the corpus.

With these observations in mind, we developed three algorithms that use information extraction as a basis for classifying texts. For each algorithm, a document is first processed by CIRCUS which generates a set of instantiated concept nodes as the representation of the text. These concept nodes are then given as input to the text classification algorithm.

4.3 The Relevancy Signatures Algorithm

Although keywords are useful as a first approximation for discriminating between relevant and irrelevant texts, they do not capture natural language context surrounding a word. Although some words are good indicators of relevance in almost any context, other words are good indicators of relevance only in specific contexts. For example, the word "dead" is a common word in the MUC-4 corpus but it is not used exclusively in relevant texts. Many texts in the MUC-4 corpus describe military incidents that are not terrorist in nature. For example, phrases involving the word "dead" frequently refer to military casualties such as: "the attack left 15 dead", or "there were 49 dead and 50 wounded". On the other hand, certain expressions involving the word "dead" are highly indicative of terrorism in the MUC-4 corpus. For example, the expression "was found dead" has an implicit connotation of foul play which often implies terrorist activity, especially in Latin American countries. In fact, every occurrence of "was found dead" in the MUC-4 corpus appears in a relevant text. Therefore the word "dead" is not a good keyword by itself, but it *is* useful for recognizing relevant texts when it appears in certain expressions.

We see a similar phenomena associated with the word "casualties". The word "casualties" is often used in military event descriptions and is therefore not a good keyword for terrorism. However, certain linguistic expressions involving the word "casualties" are good indicators of relevance for terrorism. For example, the phrase "no casualties" is often used in terrorist event descriptions to inform the reader that there were no *civilian* casualties in an attack. When we collect statistics for these two expressions in the MUC-4 corpus, we find that only 41% of the texts that contain the word "casualties" alone are relevant, but 81% of the texts that contain the expression "no casualties" are relevant. Clearly, the word "casualties" by itself is not a good keyword for terrorism but the phrase "no casualties" *is* useful for identifying relevant texts.

IR researchers have experimented with phrase-based indexing approaches that use word proximity, text structure, syntactic information and frequency data to approximately recognize phrases (e.g., [Dillon, 1983, Fagan, 1989]). But natural language processing capabilities can recognize phrases in a more robust fashion by recognizing syntactic relationships, such as active and passive verb constructions, conjunctions, prepositional phrases, etc. Fagan [Fagan, 1989] concluded that syntactic analysis would have allowed his system to produce better quality phrase descriptors than those produced by frequency and cooccurrence information alone.

The *relevancy signatures algorithm* [Riloff and Lehnert, 1992] was our first attempt to use natural language processing to classify texts on the basis of linguistic expressions instead of isolated keywords. This algorithm represents linguistic expressions as "signatures", uses statistical techniques to identify signatures that are highly correlated with relevant documents, and then uses these signatures to classify new texts.

4.3.1 Relevancy Signatures

A *signature* is a pair consisting of a word and a concept node that it triggers, which together represent a set of linguistic expressions. For example, consider the signature <murdered, $murder-passive$>. The word "murdered" triggers the concept node $murder-passive$ which is activated only when the verb "murdered" appears in a passive construction. Together, the pair represents all passive constructions of the verb "murdered", such as "was murdered", "were murdered", "have been murdered", etc. Using this representation, we can distinguish between different linguistic expressions involving the same word. For example, the signature <dead, $found-dead-passive$>

represents expressions such as "was found dead" but the signature <dead, $left-dead$> represents expressions such as "left 23 dead".

A *relevancy signature* is a signature that is highly correlated with relevance for a domain. If a new text contains a relevancy signature then, by definition, it contains a linguistic expression that is highly correlated with relevance for the domain. In the next section, we describe how to generate a good set of relevancy signatures using a training corpus and how to use them to classify new texts.

4.3.2 The Algorithm

The *relevancy signatures algorithm* has two parts: a training phase and a classification phase. During training, we generate a set of relevancy signatures based on a training corpus. During classification, we use the relevancy signatures as indices to classify new texts. Figure 4.1 shows the steps involved in the training phase.

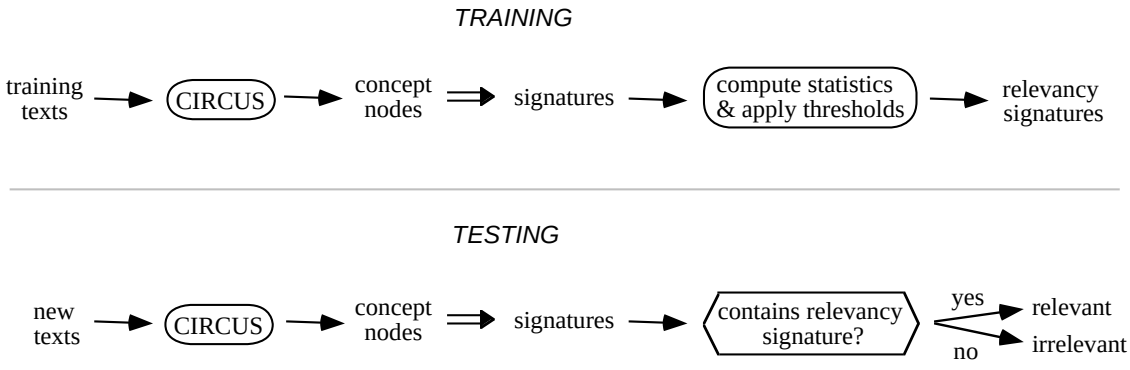


Figure 4.1: Flowchart for the relevancy signatures algorithm

Each text in the training set is first processed by CIRCUS which produces a set of instantiated concept nodes. For each instantiated concept node, a signature is created by pairing the concept node with the word that triggered it. Then statistics are compiled for each signature to determine how often it appeared in a relevant text; for each signature we estimate the conditional probability that a text is *relevant* given that it contains the signature. The formula is:

$$\Pr(\frac{text\ is\ relevant}{text\ contains\ sig_i}) = \frac{N_{sig_i \epsilon REL-TEXTS}}{N_{sig_i}}$$

where N_{sig_i} is the number of occurrences of the signature sig_i in the training set and $N_{sig_i \epsilon REL-TEXTS}$ is the number of occurrences of the signature sig_i in relevant texts in the training set. The epsilon is used loosely to denote the occurrences of the signature that "appeared in" relevant texts. Table 4.1 shows twelve signatures, their estimated conditional probabilities based on a training set of 1500 texts, and examples of sentences that generate the signatures.

The probabilities in Table 4.1 are not always intuitive. For example, 84% of the texts containing the word "assassination" were relevant but only 49% of the texts containing the word "assassinations" were relevant. On the surface, it would seem that singular and plural forms of the same word should be equally useful as indexing terms. However, this is not necessarily the case. In the MUC-4 domain, a text is relevant only if it describes a specific terrorist incident. The

Table 4.1: Sample signatures and conditional probabilities

Signature	**Prob.**	**Examples**
<assassination, $murder$>	.84	the assassination of Hector Oqueli
<assassinations, $murder$>	.49	there were 2,978 assassinations in 1988
<bombed, $bombing-passive$>	.80	public buildings were bombed
<bombed, $bombing-active$>	.51	terrorists bombed two facilities
<casualties, $no-injury$>	.81	the attack resulted in no casualties
<casualties, $injury$>	.41	the officer reported 17 casualties
<dead, $found-dead-passive$>	1.00	the mayor was found dead
<dead, $left-dead$>	.61	the attack left 9 people dead
<dead, $number-dead$>	.47	the army sustained 9 dead
<fire, $arson$>	1.00	terrorists set a restaurant on fire
<fire, $shooting$>	.87	the guerrillas opened fire
<fire, $weapon$>	.59	two helicopters were hit by rifle fire

singular form, "assassination", often reports a specific assassination of a person or group of people. But the plural form, "assassinations", is often refers to assassinations in general, e.g., "The FMLN has claimed responsibility for many kidnappings and assassinations".

Table 4.1 also shows a surprising result for the word "bombed". The passive form of the verb "bombed" is more highly correlated with relevant texts than the active form. When we look through the MUC-4 corpus for an explanation, we find that the active verb form is often used in military event descriptions but the passive form is more common in terrorist event descriptions. These distinctions would be difficult if not impossible for a person to anticipate. One of the main advantages of the corpus-based approach is that these distinctions are identified automatically using statistics from a training corpus.

To select a set of relevancy signatures, we use two thresholds: R and M. A *relevancy signature* is defined as a signature that appears at least M times in the training corpus and has conditional probability $\geq$ R. The relevancy threshold R ensures that a signature is selected as a relevancy signature only if it is highly correlated with relevance. For example, R = .85 specifies that at least 85% of the occurrences of the signature in the training set came from relevant texts. Consequently, if the signature appears in a new text then the new text is likely to be relevant. The frequency threshold M ensures a signature is not considered to be "reliable" unless it has been seen it at least M times. For example, if a signature appears only once in the training set then we do not have enough evidence to make any assumptions (positive or negative) about its general utility.

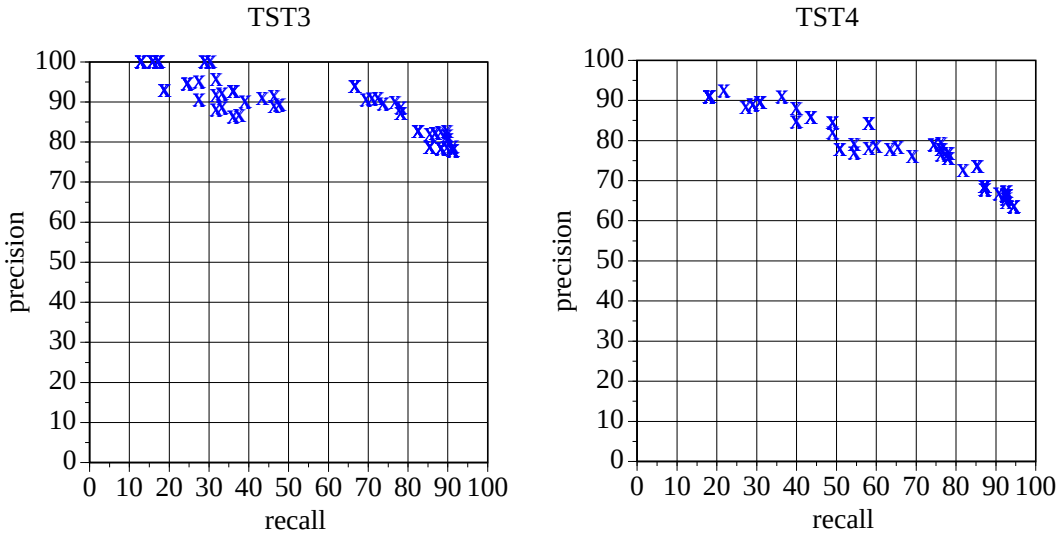
Both thresholds are inputs specified by a user. By adjusting the thresholds, the user can manipulate a tradeoff. Increasing R and M tightens the criteria for reliability and fewer signatures will be labeled as relevancy signatures. As a result, fewer texts will be classified as relevant. However, the relevancy signatures are presumably very dependable so the resulting classifications are likely to be accurate. On the other hand, decreasing R and M loosens the criteria for reliability and more signatures will be labeled as relevancy signatures. Although more texts will be classified as relevant, more false hits are also likely.

The second step of the algorithm is the classification phase shown in Figure 4.1. Given a new text to classify, CIRCUS processes the text and generates a set of instantiated concept nodes. Then signatures are created by pairing the concept nodes with their trigger words. If any of the signatures is a *relevancy signature* then the text is classified as relevant. Otherwise, the text is classified as irrelevant. An important aspect of this algorithm is that the presence of a *single* relevancy signature is enough to classify a text as relevant.

4.3.3 Experimental Results

To evaluate the performance of the relevancy signatures algorithm, we used the 1500 texts from the MUC-4 development corpus for training[4] and set aside the remaining 200 texts for testing. These 200 texts consist of two sets of 100 texts each, TST3 and TST4, which were used for the final MUC-4 evaluation and were therefore blind with respect to both the UMass/MUC-4 system and the text classification algorithms. First, each text in the training set was processed by CIRCUS and statistics were compiled for each signature. Next, the algorithm was tested on the two test sets, TST3 and TST4. Since the relevancy signatures algorithm depends on two thresholds, R and M, we tried a variety of threshold settings. We varied R from .70 to .95 in increments of .05, and we varied M from 0 to 20 in increments of 1.[5] Therefore we ran the algorithm 126 times on each test set. In Section 4.6.2 we describe a more comprehensive set of experiments that addresses the problem of how to find good threshold values empirically.

Figure 4.2 shows the scatterplots for TST3 and TST4. Each data point represents one application of the algorithm using a specific set of threshold values. Each scatterplot therefore contains 126 data points, but different threshold settings often produce the same results so many of the data points collapsed into a single point on the graph. We evaluated the algorithm on the basis of recall and precision. In general, the data points toward the right side of the graphs



Test Set	Recall	F(2)	F(1)	F(.5)	F(.3)	F(.2)	Precision
TST3	91 79	91 79	90 83	77 90	67 94	67 94	30 100
TST4	95 63	93 67	85 73	76 79	58 84	36 91	24 93

Figure 4.2: Relevancy signatures results on TST3 and TST4

[4]These were the DEV, TST1, and TST2 texts.

[5]The range of threshold values was based on our experience with the algorithm and the corpus, but the values are admittedly arbitrary.

correspond to threshold settings with lower values of R and M. The most obvious pattern in these graphs is the recall/precision tradeoff. As R and M increase, we sacrifice recall in exchange for better precision. With lower thresholds, we get high recall of over 90% but only modest precision (79% at the highest recall setting for TST3 and 63% at the highest recall setting for TST4). But it is important to interpret the precision results with respect to the number of relevant texts in each test set. Although each test set contains 100 texts, TST3 contains 69 relevant texts and TST4 contains only 55 relevant texts. These numbers represent a baseline against which precision should be assessed. For example, a constant algorithm that classifies every text as relevant will achieve 69% precision on TST3 and 55% precision on TST4. Consequently, the precision levels at the high recall end are not terribly impressive since they are only slightly above this baseline. On the left side of the graphs, the data points correspond to higher threshold values and higher levels of precision. The algorithm achieves 100% precision with 30% recall for TST3 and 93% precision with 24% recall for TST4. In between these extremes, both graphs show many data points that achieve over 80% precision with up to 50% recall.

Figure 4.2 shows a table with some of the "highlights" from the graphs. The columns display the best recall and precision scores with respect to different metrics. The column labeled **Recall** shows the scores corresponding to the data point that achieved the highest recall. Similarly, the column labeled **Precision** contains the scores for the data point that achieved the highest precision. The **Recall** and **Precision** columns represent the extreme ends of the spectrum, but it is also interesting to look at data points in between. The F-measure combines recall and precision into a single measure and accepts a β-value to adjust the relative weighting of recall and precision (the formula for the F-measure appears in Section 3.5.1). For example, β=1.0 gives recall and precision equal weighting, β=0.5 makes recall half as important as precision, and β=2.0 makes recall twice as important as precision. For each of the test sets, we identified the data points that produced the best F-measures using 5 different values of β: 2.0, 1.0, 0.5, 0.3, 0.2. These beta values cover a spectrum from highly weighted recall ($\beta = 2.0$) to strongly weighted precision ($\beta = 0.2$). Since the algorithms focus on high precision, we are mostly interested in the latter end of the spectrum. Figure 4.2 shows that relevancy signatures achieve good performance on TST3 across the board. The algorithm achieved 100% precision on TST3 with 30% recall and still achieved 94% precision at 67% recall. Relevancy signatures also get high precision ($> 90\%$) on TST4, but only at lower recall values. These results imply that relevancy signatures can be effective at high-precision text classification but the thresholds may have to be set fairly high to consistently achieve good precision across different test sets. As a result, consistent high precision may be possible only at relatively low recall levels.

4.3.4 A Simple Word-Based Algorithm

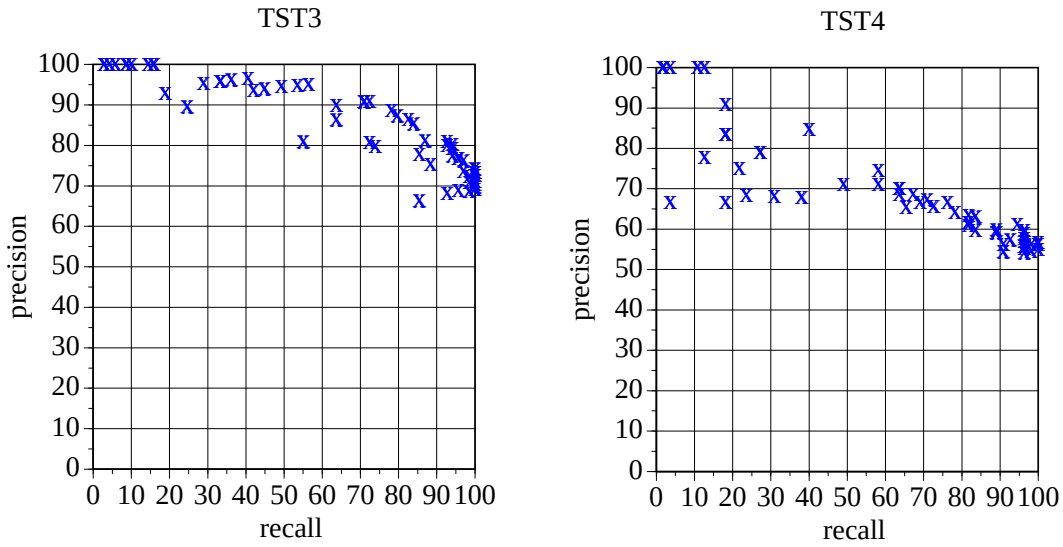
Relevancy signatures performed well on TST3 and TST4, but it is possible that a word-based approach would do just as well, or perhaps better. To address this question, we tested a simple algorithm that uses single words to classify texts. This algorithm is similar to the relevancy signatures algorithm except that the statistics are compiled for individual words instead of signatures. For each word[6] that appears in the training set, the algorithm counts how many times it appears in the training set and how often it appears in *relevant* texts in the training set. Then it estimates the conditional probability that a text is relevant given that it contains the word. The formula is:

---

[6]To give this algorithm the same advantages that the relevancy signatures algorithm had, we ran each text through CIRCUS' preprocessor first. Among other things, the preprocessor normalizes date expressions and incorporates a small phrasal lexicon so that lexicalized expressions are treated as single words.

$$\Pr(\frac{text\ is\ relevant}{text\ contains\ word_i}) = \frac{N_{word_i \epsilon REL-TEXTS}}{N_{word_i}}$$

where N_{word_i} is the number of times that $word_i$ appears in the training set and $N_{word_i \epsilon REL-TEXTS}$ is the number of times that $word_i$ appears in relevant texts in the training set. Two thresholds, R and M, are used to identify the words that are most highly correlated with relevance. A word is considered to be a *relevant word* if its conditional probability is $\geq$ R and it appeared at least M times in the training set. Finally, a new text is classified as relevant if it contains any of the *relevant words*. Note that a text will be classified as relevant even if it contains only a single word that is highly correlated with relevance.



Test Set	Recall	F(2)	F(1)	F(.5)	F(.3)	F(.2)	Precision
TST3	100 74	100 74	94 80	72 91	57 95	57 95	16 100
TST4	100 57	100 57	95 61	58 74	40 85	40 85	13 100

Figure 4.3: Simple keyword algorithm on TST3 and TST4

We tested this algorithm using the same training and test sets as before. We varied R from .70 to .95 in increments of .05, and varied M from 0 to 50 in increments of 5. Figure 4.3 shows the results of this algorithm on TST3 and TST4. The *relevant words algorithm* performs quite well on TST3, particularly at the high precision end where it consistently achieves $\geq$ 90% precision for recall levels under 50%. However, precision falls quickly at the high recall end, down to 69% at 100% recall[7] and even below 69% at other points. Remember that 69% precision is the baseline for TST3, so the algorithm is classifying *every* text as relevant for the data point at 100% recall, 69% precision.

On TST4, the word-based algorithm has much more difficulty. Precision levels are low across the board except at the extreme low recall end. In fact, there are data points at the low recall

[7]There are multiple data points at 100% recall.

end with both high and low precision. The data points are so scattered because the algorithm is classifying very few texts as relevant. For example, the leftmost data point toward the bottom of the TST4 graph corresponds to 3.6% recall and 67% precision. But 3.6% recall of 55 relevant texts means that the algorithm correctly classified only 2 relevant texts. Since the precision is 67%, the algorithm must have classified a total of 3 texts as relevant. When this single misclassified text is correctly classified the precision jumps to 100%. At low recall levels, changing the classification of a single text can have a dramatic impact on precision and, as a result, precision is extremely volatile. We conclude that this simple word-based algorithm can achieve good performance on some texts but cannot consistently obtain high precision. The difference between this word-based approach and the IE-based approaches will become more pronounced in the next two sections.

4.4 The Augmented Relevancy Signatures Algorithm

Relevancy signatures identify key phrases and expressions that are strongly associated with relevance for a domain. However, they are susceptible to false hits when a key phrase occurs in an irrelevant context. For example, consider the following two sentences:

(a) A car bomb exploded.
(b) The foreign debt crisis exploded.

Both of these sentences are represented by the signature <exploded, $explosion$>. But (a) describes a terrorist event and (b) does not. Metaphorical expressions are pervasive in language and can cause false hits during text classification. Expressions like "killing the agreement", "death to communism", and "an attack on freedom" are prevalent in the MUC-4 corpus.

Relevancy signatures can fail when a correct classification depends on additional context surrounding a phrase. Even without metaphorical language, contextual distinctions can be a common source of false hits. For example, consider these two sentences:

(a) The peasants were attacked by the rebels.
(b) Kent Jr. was attacked by three other Pavon Prison inmates.

Once again, both sentences are represented by the same signature <attacked, $attack-passive$> but (a) describes a terrorist incident and (b) does not. The identity of the perpetrator (rebels vs. inmates) is critical in distinguishing a terrorist event from a non-terrorist event. To address these problems, we extended the relevancy signatures algorithm to include slot filler information. By augmenting the signatures with slot fillers, we capture local context surrounding the key phrase which can improve the accuracy of the resulting classifications.

4.4.1 Augmented Relevancy Signatures

A relevancy signature represents the presence of a key phrase in a text. By representing only the *existence* of a concept node, it ignores the surrounding context that is available inside the concept node. Augmented relevancy signatures [Riloff and Lehnert, 1992] use the information extracted by the concept nodes in combination with the signatures to classify texts. *Relevancy signatures represent the existence of concept nodes; augmented relevancy signatures represent concept node instantiations.*

Each slot filler is represented as a triple of the form: (concept node type, slot name, semantic feature). For example, suppose a kidnapping concept node successfully extracts the victim "the mayor of Achi". The victim yields the following *slot triple*: (kidnapping, victim, GOVERNMENT-OFFICIAL), because the word "mayor" is tagged with the semantic feature GOVERNMENT OFFICIAL

in the dictionary.[8] This slot triple represents the fact that a government official was identified as the victim of a kidnapping event. We use semantic features instead of lexical items to generalize over the specific words that appeared in the text.

An *augmented relevancy signature* is the combination of a signature and a slot triple that are both highly correlated with relevance for a domain, independently. If a text contains an augmented relevancy signature then it contains both a highly relevant key phrase and a highly relevant piece of information surrounding the key phrase. The criteria for augmented relevancy signatures are strict because both sources of information must be highly correlated with relevance independently. As a result, augmented relevancy signatures are very effective at classifying texts accurately.

4.4.2 The Algorithm

The augmented relevancy signatures algorithm is the same as the relevancy signatures algorithm except that statistics are collected for slot triples as well as signatures. For each concept node produced by CIRCUS, a signature and a set of slot triples are created. For each slot triple, the algorithm estimates the conditional probability that a text is relevant given that the slot triple appears in the text. Finally, "reliable" slot triples are identified by using two thresholds that are analogous to the relevancy signature thresholds: R_{slot} and M_{slot}. A slot triple is judged to be "reliable" if it appears at least M_{slot} times in the training corpus and if its conditional probability is $\geq R_{slot}$. Figure 4.4 illustrates the training procedure.

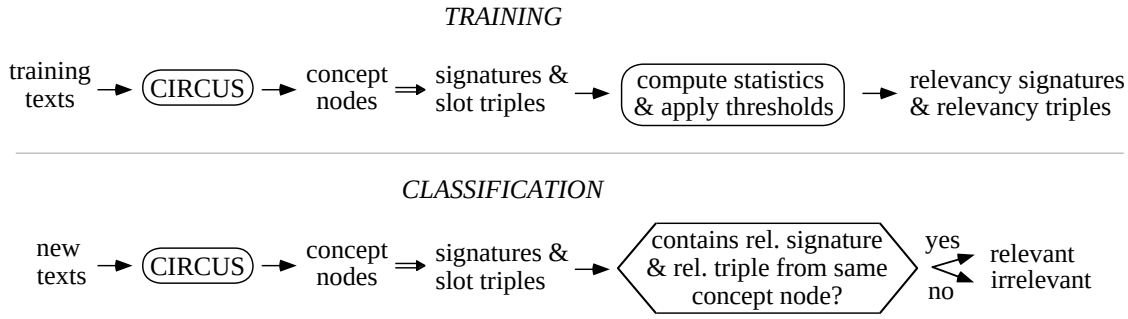


Figure 4.4: Flowchart for the augmented relevancy signatures algorithm

To classify a new text, CIRCUS processes the text and generates a signature and set of slot triples for each concept node. If any concept node yields both a relevancy signature and a reliable slot triple then the text is classified as relevant. Otherwise, the text is classified as irrelevant.

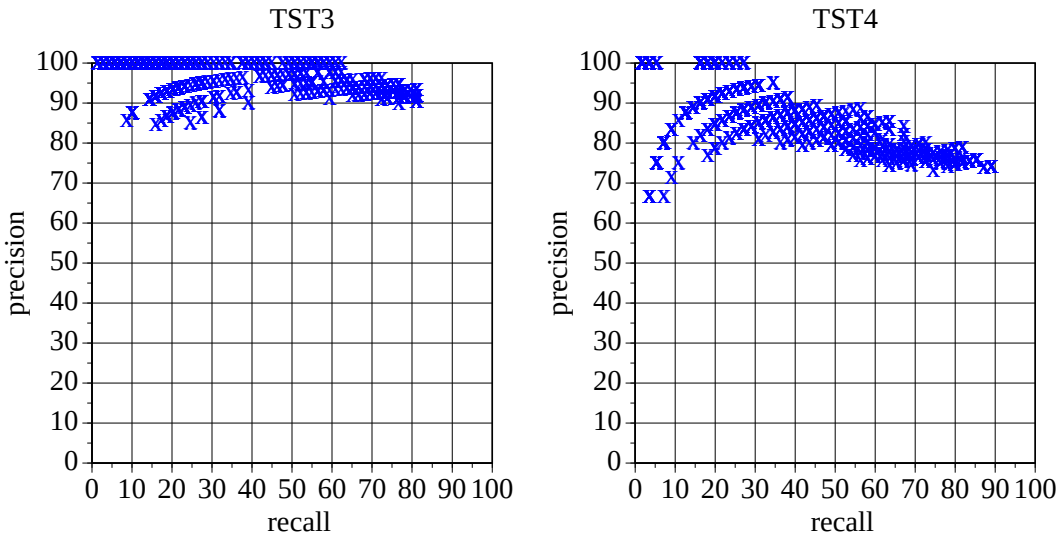
4.4.3 Experimental Results

We evaluated the augmented relevancy signatures algorithm on the basis of the same test sets, TST3 and TST4. First, we used the training set of 1500 texts to generate signature and slot triple statistics. Once again, we tested the algorithm with a variety of threshold settings. We varied

[8]Some words have multiple semantic features assigned to them. In this case, we create multiple slot triples for a filler, one for each semantic feature.

each of R and R_{slot} from .70 to .95 in increments of .05, and each of M and M_{slot} from 0 to 20 in increments of 5.[9]

Figure 4.5 shows the scatterplots for the augmented relevancy signatures algorithm on TST3 and TST4. Each data point represents the application of the algorithm using one set of threshold values. As before, many different threshold settings produce identical results so we see far fewer than 900 data points. The recall/precision tradeoff is still apparent in these graphs but is more difficult to see because the combinations of the thresholds are more complex. The tails on the left side of the graph are caused by low recall which makes precision especially volatile. As we noted in Section 4.3.4, at very low recall values changing the classification of a single text can have a dramatic impact on precision. The augmented relevancy signatures algorithm is especially susceptible to these effects because its strict criteria for relevance can result in relatively few relevant classifications.



Test Set	Recall	F(2)	F(1)	F(.5)	F(.3)	F(.2)	Precision
TST3	81 93	81 93	81 93	81 93	62 100	62 100	62 100
TST4	89 74	89 74	89 74	67 84	56 89	27 100	27 100

Figure 4.5: Augmented relevancy signatures results on TST3 and TST4

We see several differences between these results and the results for the relevancy signatures algorithm. The most striking difference is that augmented relevancy signatures achieve better performance on TST3. Augmented relevancy signatures reach 100% precision at 62% recall, but relevancy signatures reach 100% precision at only 30% recall. The augmented relevancy signatures algorithm also achieves 100% precision with 27% recall on TST4 where relevancy signatures alone could achieve only 93% precision with 24% recall on TST4.

[9]We used bigger increments for M than in the previous experiments because the combinatorics of varying *four* thresholds can quickly get out of hand. Even with the larger increments of 5, we ran the algorithm 900 times on each test set.

On the surface, it might seem counterintuitive that the algorithm with stricter criteria for relevance produces better recall (at 100% precision) than the algorithm with less strict criteria. The reason is subtle. The relevancy signatures algorithm classifies a text as relevant if it contains a single key phrase that is highly correlated with relevance. As a result, it may be able to attain high precision only with high threshold values. The augmented relevancy signatures algorithm classifies a text as relevant if it contains a key phrase and a piece of extracted information that are both highly correlated with relevance. Therefore, it can often sustain high precision levels with lower threshold values. The low threshold values enable the algorithm to achieve better recall levels while maintaining strong precision.

It is also worth noting that augmented relevancy signatures perform well on TST3 at the high recall end, achieving 93% precision with 81% recall. Both algorithms have difficulty on TST4 at the high recall end, but augmented relevancy signatures obtain better precision (74%) than relevancy signatures alone (63%) without sacrificing much recall.

4.5 Case-based Text Classification

As we noted earlier, keyword approaches in general and relevancy signatures in particular are prone to false hits when the correct classification of a text depends on the context surrounding a word or phrase. Augmented relevancy signatures were the first attempt to go beyond keywords and phrases and take advantage of local context. Augmented relevancy signatures demonstrated good success with high-precision text classification, but sometimes only with moderate recall.

Many texts clearly describe a relevant terrorist incident even though they do not contain any specific words or phrases that are strongly associated with relevance. Some sentences contain multiple pieces of information that are relevant *together* but are not necessarily relevant independently. For example, the word "killed" is not highly correlated with terrorism because people are killed in many situations that are not terrorist in nature. However, if a MUC-4 text mentions that a government official was killed then the text probably is describing a terrorist incident because government officials are frequently the victims of terrorist attacks in Latin America. Even so, many texts that mention a government official are not relevant. To reliably classify texts, we need to consider both pieces of information together. In this situation, a weak key phrase ("was killed") identifies a potentially relevant event and a strong slot filler (government official victim) provides additional evidence that the event is probably terrorist.

A strong key phrase combined with a weak slot filler can be equally effective. For example, the word "assassination" is an important word in the domain of terrorism, but is not always reliable because many irrelevant texts in the MUC-4 corpus refer to assassination in general. The presence of a known victim often distinguishes a general reference to assassination from a specific one. Since the word "assassination" is such a strong cue for terrorism, just about any reference to a victim will do. For example, if a text describes the assassination of an individual then it probably describes a specific terrorist attack. Once again, the key phrase and the slot filler are not highly correlated with relevance independently so both pieces of information are needed to make a reliable classification. The presence of a strong key phrase ("assassination") identifies a potentially relevant event and the slot filler (any victim) serves as an additional source of evidence that the text is relevant.

However, some relevant texts do not contain any strong key phrases or strong slot fillers. Sometimes, a text merely contains an abundance of information that, in total, describes a relevant incident. In these texts, the whole is more compelling than the parts. For example, consider the following sentences:

> Police sources have confirmed that a guerrilla was killed and two civilians were wounded this morning during an attack by urban guerrillas.
>
> The mayor reiterated his position when he commented on the attack in which 20 persons were killed and approximately 100 were injured, which was perpetrated yesterday by terrorists on the drug cartel's payroll near Itagui municipality.

> Two vehicles were destroyed and an unidentified office of the agriculture and livestock ministry was heavily damaged following the explosion of two bombs yesterday afternoon.

Each of these sentences clearly describes a specific terrorist incident. However, none of the words or phrases are necessarily relevant on their own. In the first sentence, the words "killed", "wounded", and "attack" all describe a violent incident but could easily refer to a military incident. The phrase "urban guerrillas" is certainly associated with terrorism but guerrillas are mentioned in many texts that describe military incidents or do not mention any specific incidents at all. The bottom line is that sometimes we need *multiple* pieces of information to conclude that a text is relevant. For the terrorism domain, there must be evidence of a violent act (e.g., "attack") perpetrated by terrorists (e.g., "by urban guerrillas") against civilian targets (e.g., "two civilians were wounded"). This information is compelling *collectively* and we need all of it to confidently classify the text as relevant.

In this section, we describe a text classification algorithm that uses case-based reasoning to classify texts. Case-based reasoning techniques use the solutions to previous problems (called "cases") to solve new ones (e.g., [Ashley, 1990, Hammond, 1986, Kolodner and Simpson, 1989]). By representing natural language contexts as cases, information that spans multiple clauses is used collectively to classify texts.

4.5.1 The Case Representation

Ideally, a text should be classified as relevant or irrelevant on the basis of the entire document. However, to avoid the problems of discourse analysis[10], we used single sentences as a first approximation toward larger contexts. Each case represents the natural language context associated with a single sentence.

To create a set of cases for a document, for each sentence we collect all of the concept nodes produced by CIRCUS and merge them into a case. A case is represented as a structure with five slots: *signatures*, *perpetrators*, *victims*, *targets*, and *instruments*. The *signatures* slot contains the signatures associated with each concept node generated by the sentence. The remaining four slots contain the information corresponding to the fillers that were extracted by these concept nodes.[11] The concept nodes extract specific strings from the text (e.g., "the mayor"), but only the semantic features associated with the strings are stored in the case (e.g., GOVERNMENT-OFFICIAL). Figure 4.6 shows a sample sentence, the concept nodes produced by CIRCUS for the sentence, and the resulting case representation.

Note that the case representation does not preserve the associations between concept nodes and their fillers. For example, the case in Figure 4.6 does not specify whether the GOVT-OFFICE-OR-RESIDENCE was destroyed and the TRANSPORT-VEHICLE was damaged or vice versa. We purposely disassociated the slot fillers from the concept nodes that extracted them so that the algorithm can search for relationships between any signature and filler.

4.5.2 The Case Base

Each document is represented as a set of cases, one for each sentence that produced at least one concept node. During training, each text in the training corpus is converted into a set of

---

[10]For the MUC task, discourse analysis refers to the problem of tracking multiple event descriptions in a single text. Discourse analysis was one of the most difficult problems encountered in MUC-3 [Iwanska *et al.*, 1991].

[11]The concept nodes in the MUC-4 dictionary extract only four classes of information. In general, a case should have one slot for each concept node slot defined for the domain.

SENTENCE: Two vehicles were destroyed and an unidentified office of the agriculture and livestock ministry was heavily damaged following the explosion of two bombs yesterday afternoon.

CONCEPT NODES

$destruction-passive$ (triggered by "destroyed")
target = two vehicles

$damage-passive$ (triggered by "damaged")
target = an unidentified office of the agriculture and livestock ministry

$weapon-bomb$ (triggered by "bombs")

CASE

Signatures: (<destroyed, $destruction-passive$>,
<damaged, $damage-passive$>,
<bombs, $weapon-bomb$>)
Perpetrators: nil
Victims: nil
Targets: (GOVT-OFFICE-OR-RESIDENCE TRANSPORT-VEHICLE)
Instruments: (BOMB)

Figure 4.6: A sample sentence, concept nodes, and resulting case

cases which are stored in a case base. The resulting case base contains thousands of natural language contexts from hundreds of texts. It is important to note that *the case base is constructed automatically as a side effect of natural language processing.*

To classify a new text, the text is converted into cases and the algorithm determines whether any of its cases are relevant to the domain. The heart of the algorithm is its ability to accurately judge the relevancy of new cases. If any of the cases are deemed to be relevant then the text is classified as relevant, otherwise it is classified as irrelevant.

To determine the relevancy of a new case, the most obvious approach would be to retrieve the most similar case from the case base and apply its classification to the new case. Most case-based reasoning (CBR) systems retrieve one or a few similar cases and apply them directly to the current case (e.g., [Ashley, 1990, Hammond, 1986]). We cannot do this, however, because the MUC-4 corpus provides us with the correct classifications for each *document* but not for each *case*. If a document is irrelevant, then the text does not contain any relevant information so all of its cases must be irrelevant. However, if a document is relevant then some of the sentences in the text describe a relevant incident but we do not know which ones. This is a classic example of the credit assignment problem.[12] We do not know which cases contain the information that is responsible for the relevant classification of the document.

To get around this problem, the algorithm relies on statistics to sift through the case base and identify cases that probably contain relevant information. The general approach is similar to that of the previous algorithms. The algorithm probes the case base with a set of features, retrieves cases that share these features, and looks at the statistical properties of the retrieved cases. If a

[12]The *credit assignment problem* is a well-known term used by artificial intelligence researchers. It refers to the difficulty of determining which part of a system deserves credit (blame) for a correct (incorrect) result.

high percentage of the retrieved cases come from relevant documents then it assumes that this is not a coincidence. The retrieved cases must share something in common that makes them relevant. Since the cases all share the probe features, these features probably represent relevant information. It follows that new cases containing these features are also likely to be relevant.

Ideally, we would like to retrieve all cases from the case base that share exactly the same features as the new case. However, the case representation is rich enough and the training corpus is not large enough for the case base to contain many exact matches. Since the algorithm relies on the statistical properties of the retrieved cases to determine whether the cases are truly relevant, it must retrieve a reasonably large number of cases. So instead of looking for exact matches, it uses *relevancy indices* to retrieve cases that share a few specific features. In the next section, we introduce the notion of a *relevancy index*, describe its representation, and explain how it is used to retrieve cases.

4.5.3 Relevancy Indices

A *relevancy index* is a collection of features that, together, reliably predict a relevant event description. To make things less abstract, we first describe the representation and then justify it with examples. A relevancy index is a triple of the form: (signature, slot filler, case outline). As we have already explained, a signature represents a set of linguistic expressions. A slot filler is a pair consisting of the name of a slot and a semantic feature representing the information extracted by the slot, such as: (perpetrators, TERRORIST).[13] The third part of a relevancy index is the *case outline*. An outline is a list of slots that contain fillers. For example, the outline (perpetrators, victims) represents a case that contains information in the perpetrator and victim slots but not in the target or instrument slots. The signature slot is always filled so it is not included as part of the outline. The case outline represents the *types* of information that appear in a sentence.

By combining a signature, slot filler, and case outline into a single index, the algorithm retrieves cases that share similar key phrases, at least one piece of similar extracted information, and that contain roughly the same amount and types of information. By indexing simultaneously on a signature and a slot filler, it retrieves cases that are strongly associated with terrorism because the combination of a key phrase and slot filler is compelling. The relevancy index may represent a weak key phrase and a strong slot filler, a strong slot filler and a weak key phrase, or a weak key phrase and a weak slot filler. However, a high percentage of the cases retrieved by the index will be relevant only if the two items together are highly associated with terrorism.

The case outline captures the amount of information in a sentence and the types of information. Intuitively, we included the case outline because different signatures and slot fillers require different amounts and types of supporting information in order to be reliable. For example, consider the three relevancy indices in Table 4.2.

The first index retrieves cases that contain the word "assassination", a civilian victim, and no additional information. We retrieved all cases in a case base derived from 1500 texts that share this index: 100% of the retrieved cases came from relevant texts. This is not surprising since the word "assassination" is a strong keyword for terrorism, especially when paired with a specific victim. However, when we probed the same case base using the second relevancy index, we find that only 68% of the retrieved cases came from relevant texts. The only difference between these indices is the signature. Once again, this is not surprising since the word "killed" is not a good keyword for terrorism in the MUC-4 corpus. In particular, the corpus contains many texts that

[13]Note that these are different from the slot triples used for augmented relevancy signatures. In the augmented relevancy signatures algorithm, the statistics for slot triples are computed separately from the signatures. We dropped the event type (e.g., murder) from the relevancy indices because they already include a signature (which represents a concept node and therefore an event type).

Table 4.2: The power of the case outline

Relevancy Index	Rel. Cases
(<assassination, $murder$>, (victims, CIVILIAN), (victims))	100%
(<killed, $murder-passive$>, (victims, CIVILIAN), (victims))	68%
(<killed, $murder-passive$>, (victims, CIVILIAN), (victims perpetrators))	100%

contain summary descriptions of terrorist activity over a period of time, such as: many people have been killed since 1980.

However, these summary descriptions can be distinguished from specific event descriptions merely by changing the case outline. When we probed the case base using the third relevancy index, we found that 100% of the retrieved cases came from relevant texts. The difference between the last two indices is the case outline. The second index dictates that the retrieved cases must contain only victims but the third index dictates that the retrieved cases must contain victims and perpetrators. General summary descriptions usually do not mention a perpetrator, but specific event descriptions typically do mention a perpetrator. The presence of a perpetrator can be critical in distinguishing between specific and general event descriptions. Of course, this particular distinction does not always hold. For example, a summary event description may blame a particular terrorist organization for a wave of attacks, or a specific event description may fail to identify a perpetrator. These are merely patterns that generally hold in the MUC-4 corpus. The emphasis is not on this particular example, but on the role of the case outline as a feature for exploiting these types of contextual distinctions.

4.5.4 The Algorithm

Finally, we explain how relevancy indices are used to classify new documents. The case-based text classification algorithm [Riloff, 1993b] is illustrated in Figure 4.7.

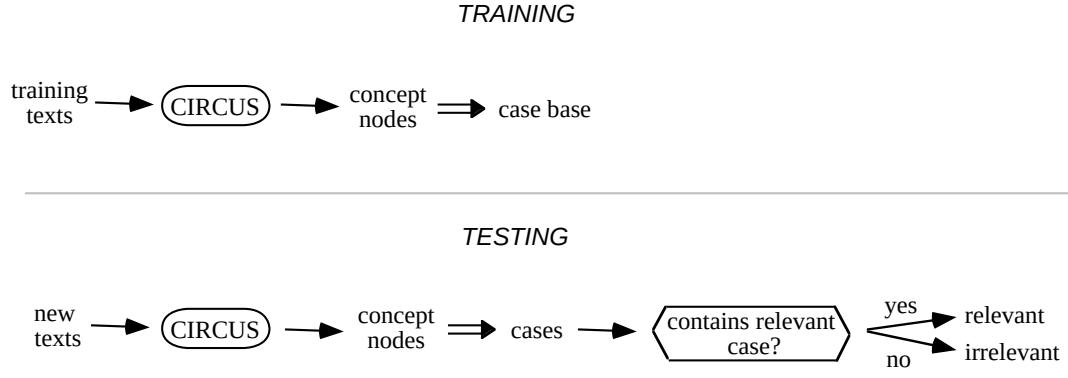


Figure 4.7: Flowchart for the case-based text classification algorithm

To classify a new document, the concept nodes produced by CIRCUS are converted into cases. If any of the cases are judged to be relevant then the text is classified as relevant, otherwise it is classified as irrelevant. A *case* is judged to be relevant if the following three conditions are satisfied:

Condition 1: The case contains a strong *relevancy index.*
Condition 2: The case does not contain any "bad" signatures.
Condition 3: The case does not contain any "bad" slot fillers.

Condition 1 is the most important part of the algorithm. Conditions 2 and 3 are merely secondary checks to make sure that the case does not contain any information that might negate otherwise relevant information. We will show examples of these situations later.

To determine whether a case contains a strong relevancy index, the algorithm generates all possible relevancy indices for the case. Most cases have multiple signatures and slot fillers so there are often many possible relevancy indices for a case.[14] For each relevancy index, the algorithm retrieves all cases from the case base that share the same index and estimates the conditional probability that a case is relevant given that it shares the index. The formula is:

$$\Pr(\frac{case\ is\ relevant}{case\ matches\ r_i}) = \frac{N_{C_{r_i} \epsilon REL-CASES}}{N_{C_{r_i}}}$$

where r_i is relevancy index i, $N_{C_{r_i}}$ is the number of retrieved cases, and $N_{C_{r_i} \epsilon REL-CASES}$ is the number of retrieved cases that come from relevant texts. If this probability is high then we assume that the relevancy index is responsible for the high correlation with relevant texts. In other words, we assume that the relevancy index represents information that is relevant to the domain. It follows that the new case, which shares the index, also contains information that is relevant to the domain and should be classified as relevant.

Two thresholds are used to determine whether the probability is high enough: a relevancy threshold, R_{cases}, and a frequency threshold, M_{cases}. If the conditional probability is $\geq R_{cases}$ and $N_{C_{r_i}} \geq M_{cases}$ then the relevancy index is deemed to be "strong" and Condition 1 is satisfied.

Conditions 2 and 3 look for "bad" signatures and slot fillers that might be negative indicators for the domain. The presence of a particular signature or slot filler may warrant an irrelevant classification even though the rest of the information seems relevant. As we mentioned earlier, the MUC-4 domain guidelines specify that a document is relevant only if it describes a *specific* terrorist incident. Summary event descriptions are not considered to be relevant, for example:

> More than 100 people have died in Peru since 1980, when the Maoist Shining Path organization began its attacks and its wave of political violence.

This sentence yields the following case:

> **Case**
> **Signatures:** (<died, die>,<wave, $generic-event-marker$>,<attacks, $attack-noun$>)
> **Perpetrators:** (TERRORIST ORGANIZATION)
> **Victims:** (HUMAN)
> **Targets:** nil
> **Instruments:** nil

However a similar sentence describes a specific, relevant incident:

> More than 100 people have died in Peru during 2 attacks by the Maoist Shining Path organization yesterday.

---

[14]This is potentially expensive but, in practice, we are constrained by the nature of language. Most sentences contain only a relatively small amount of information.

This sentence yields the following case:

Case
Signatures: (<died, die>,<attacks, $attack-noun$>)
Perpetrators: (TERRORIST ORGANIZATION)
Victims: (HUMAN)
Targets: nil
Instruments: nil

The cases for these two sentences are almost identical. The only difference is that the first sentence produces a concept node called $generic-event-marker$ in response to the phrase "wave of". The UMass/MUC-4 system used a small number of special concept nodes to recognize textual cues that signal summary event descriptions. The presence of this single concept node indicates that the sentence is describing a summary event description and is therefore irrelevant, even though the rest of the information appears to be relevant.

Similarly, a "bad" slot filler can indicate that a case is irrelevant, despite otherwise good information. For example, a terrorist attack must involve a terrorist perpetrator. If a civilian commits a crime, then it is not considered to be terrorist in nature. As a result, the sentence "A guerrilla killed the mayor" is relevant but the sentence "A burglar killed the mayor" is not.

Two additional "irrelevance" thresholds, I_{sig} and I_{slot}, are used to identify "bad" signatures and slot fillers. Given a case, each signature is checked by retrieving all cases from the case base that contain the signature estimating the conditional probability that a case is relevant given that it contains the signature. That is,

$$\Pr(\frac{case\ is\ relevant}{case\ contains\ sig_i}) = \frac{N_{C_{sig_i} \epsilon REL-CASES}}{N_{C_{sig_i}}}$$

where $N_{C_{sig_i}}$ is the total number of retrieved cases that contain sig_i and $N_{C_{sig_i} \epsilon REL-CASES}$ is the number of retrieved cases that contain sig_i that come from relevant texts. If the probability $< I_{sig}$ and $N_{C_{sig_i}} \geq M_{cases}$, then the signature does not satisfy Condition 2. The procedure for slot fillers is analogous. Given a case, each slot filler is checked by retrieving all cases from the case base that contain the slot filler and the same case outline and estimating the conditional probability that a case is relevant given that it contains the slot filler and outline. That is,

$$\Pr(\frac{case\ is\ relevant}{case\ contains\ filler_i\ and\ outline_j}) = \frac{N_{C_{slot_{ij}} \epsilon REL-CASES}}{N_{C_{slot_{ij}}}}$$

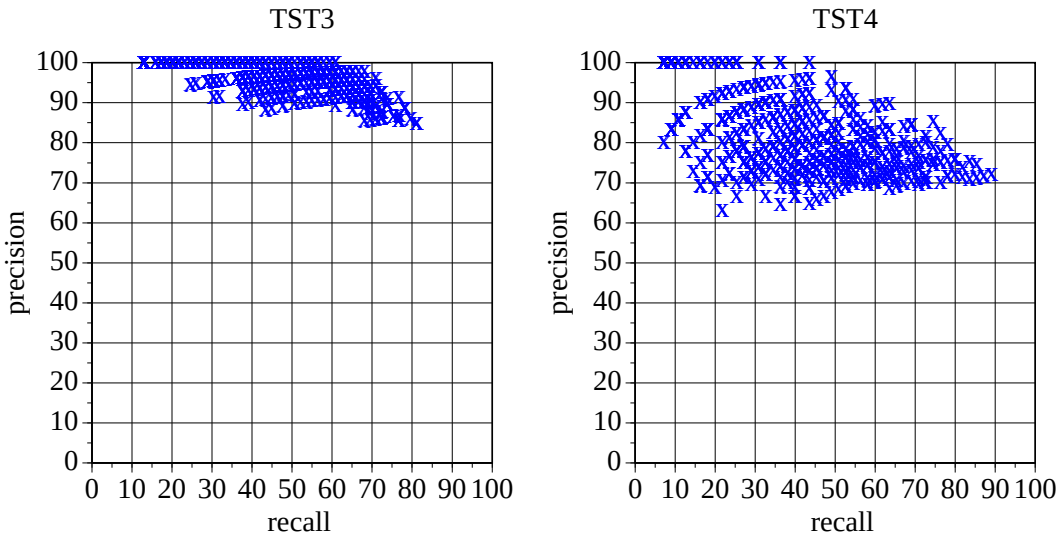
where $N_{C_{slot_{ij}}}$ is the total number of retrieved cases that contain $filler_i$ and $outline_j$ and $N_{C_{slot_{ij}} \epsilon REL-CASES}$ is the number of retrieved cases that come from relevant texts. If the probability is $< I_{slot}$ and $N_{C_{slot_{ij}}} \geq M_{cases}$, then Condition 3 is not satisfied.

These four thresholds allow the user to tailor the performance of the algorithm for the domain. By increasing I_{sig} and I_{slot} the user can adjust the sensitivity of the algorithm to irrelevance cues. The user can also adjust M_{cases} based on the size of the training corpus. For a large corpus, the user may want to give M_{cases} a large value to get more accurate probability estimates. For a smaller corpus, the user may give M_{cases} a small value because the case base contains relatively few cases. We address the issue of selecting appropriate threshold values automatically in Section 4.6.1.

4.5.5 Experimental Results

We evaluated the case-based text classification algorithm using the same training and test sets as in the previous experiments. First, we created a case base from the 1500 texts in the training set. The final case base contained 6868 cases. Next, we tested the algorithm on both TST3 and TST4 with a variety of threshold settings. We varied R_{cases} from .70 to .95 in increments of .05, M_{cases} from 0 to 20 in increments of 2, and gave each of I_{sig} and I_{slot} the values {.50, .60, .70}.

Figure 4.8 shows the results for TST3 and TST4. Once again, the data points are somewhat scattered and we see the familiar tails on the low recall side. The most notable improvement is on TST4. The case-based algorithm achieved 44% recall with 100% precision where the augmented relevancy signatures algorithm reached only 27% recall with 100% precision. These results support our claim that using natural language contexts allows the algorithm to correctly classify texts that were inaccessible to the word and phrase-based algorithms. On TST3, the results are very similar to the augmented relevancy signatures algorithm, although the case-based algorithm produces a strong data point in the F(.5) column (68% recall with 98% precision).



Test Set	Recall	F(2)	F(1)	F(.5)	F(.3)	F(.2)	Precision
TST3	81 85	81 85	77 91	68 98	61 100	61 100	61 100
TST4	89 72	89 72	89 72	75 85	44 100	44 100	44 100

Figure 4.8: Case-based text classification results

Figure 4.8 shows many data points with relatively low precision on TST4. This is because using larger natural language contexts for classification can work against you when using low threshold values. One of the main strengths of the case-based approach is that it can classify a text as relevant based on context even if the text does not contain any reliable key words or phrases. With high threshold values, a case is classified as relevant when its context is highly correlated with relevance. However, with low threshold values, a case is classified as relevant even when its context is weakly correlated with relevance. This means that the text may not contain any relevant words, phrases, or information! As a result, the case-based approach is not particularly effective with low threshold values.

Fortunately, the case-based algorithm produces similar, or better, recall levels than the other algorithms, even with higher threshold values. However, to get the best performance from the algorithm, the user needs to choose appropriate threshold settings. In the next section, we describe a procedure that allows the user to find good threshold values empirically.

4.6 Comparative Analysis in Multiple Domains

In the previous sections, we presented three text classification algorithms and showed that performance improves as additional context is used to classify texts. However, the experiments so far were based on two blind test sets of 100 texts each. It would be premature to come to any conclusions about the relative merit of the algorithms based solely on these 200 texts. With such a small number of texts, effects can be magnified (in a positive or negative light) and the texts may not be representative of the corpus in general.

Furthermore, we applied the algorithms only to the MUC-4 domain of terrorism. To make general claims about the effectiveness of the algorithms and their relative merits, we must apply them to additional domains. We also must address the issue of how to find appropriate threshold settings. In the previous experiments, we tested the algorithms using many different threshold values and presented the best results. However, in a real-world scenario, a user must choose a specific set of threshold values.

In this section, we address these issues by:

1. presenting a procedure for deriving appropriate threshold values empirically.
2. applying the text classification algorithms to a larger test set for terrorism using a cross-validation design.
3. applying the text classification algorithms to two additional domains: joint ventures and microelectronics.

4.6.1 Deriving Threshold Values Empirically

The text classification algorithms that we presented require several thresholds to select the most "reliable" indexing terms. However, we have not addressed the issue of how to find appropriate threshold values. In a new domain or corpus, a user would not know what threshold values to use. The "best" threshold settings depend on the size and nature of the corpus and, most importantly, the needs of the user. The algorithms all support a recall/precision tradeoff that can be manipulated by adjusting the thresholds. In general, a user who wants high recall should choose low threshold values and a user who demands high precision should choose high threshold values.

Before we present the procedure for identifying good threshold values, we must describe the cross-validation design that we will use to evaluate the text classification algorithms. Ideally, we would like to evaluate the algorithms on the basis of a large number of texts, but they also require a large training set and our available corpora are relatively small. Instead, we use a 10-fold cross-validation design that allows us to use *all* of the texts in the corpus for testing.

The cross-validation design uses all of the texts for testing by rotating different subsets of the corpus for training. First, we randomly divide the entire corpus of 1700 texts into 10 partitions of 170 texts each. Then we evaluate each "fold" independently. For the first fold, we use partition #1 as the test set and the remaining 9 partitions as the training set. For the second fold, we use partition #2 as the test set and the other 9 partitions as the training set, and so on. Each fold therefore has a unique test set of 170 texts and a training set of 1530 texts (the training sets will overlap).

Figure 4.9 illustrates the process for a *single* fold. We use the training set to generate statistical training data (in the form of signatures, slot triples, or a case base) and then apply the algorithm

to the corresponding test set. Finally, we combine (sum) the results over all 10 folds. For example, in a 2-fold cross-validation design, if fold #1 correctly classified 24 of 50 texts and fold #2 correctly classified 16 of 50 texts then the combined results would be 40 out of 100 texts. The combined results reflect the performance of the algorithm on all 1700 texts, albeit with different training sets.

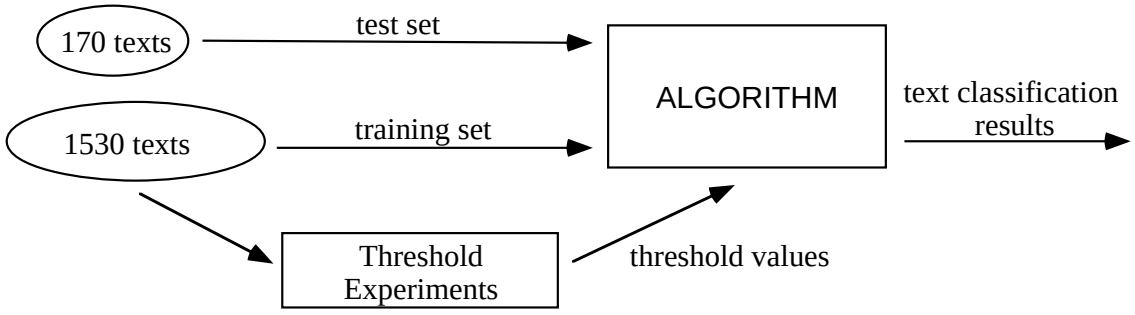


Figure 4.9: One fold of cross-validation

Since the effectiveness of the thresholds is entirely dependent on the corpus, we devised a procedure that allows the user to derive the "best" threshold settings empirically. The procedure recommends threshold settings that are appropriate for a variety of metrics. By using multiple metrics, we create a recall/precision "knob". A user can look at the results obtained on the training set under each metric and choose threshold settings that achieved the desired behavior. In this manner, users can automatically identify the most appropriate threshold values for their needs based on the corpus. The suggested threshold settings are not guaranteed to produce the best possible results but they achieved the desired behavior on training texts.

We used seven metrics to represent a spectrum that gradually shifts importance from recall to precision. At the two ends of the spectrum, the metrics represent the absolute importance of recall (i.e., the highest recall regardless of precision) and the absolute importance of precision (i.e., the highest precision regardless of recall). In between these endpoints, we use the F-measure to gradually vary the relative importance of recall and precision by using different β-values. As before, we chose five values of β: 2.0, 1.0, 0.5, 0.3, 0.2. Note that several of the β-values give extra weighting to precision because our algorithms were specifically designed for high-precision text classification.

Figure 4.10 shows the procedure for automatically deriving good threshold values. This design mimics the 10-fold cross-validation experiment. Using the same training set of 1530 texts that were used to generate statistical data, we randomly divide the training set into 10 partitions of 153 texts each. For the first fold, we use partition #1 as the test set and the remaining 9 partitions as the training set. For the second fold, we use partition #2 as the test set and the other 9 partitions as the training set, etc. For each fold, we use the training set to generate statistical data and then run the algorithm on the test set with a variety of different threshold settings.[15] Then we sum

[15]We tested the algorithms with the same variety of threshold settings as before, with a few exceptions: we varied M from 0-20 in increments of 2 for the relevancy signatures algorithm, we varied M and M_{slot} from 0-15 in increments of 5 for the augmented relevancy signatures algorithm, and we varied M_{cases} from 0-15 in increments of 5 for the case-based algorithm. The larger increments were necessary to keep the total number of runs under control during the cross-validation and threshold experiments.

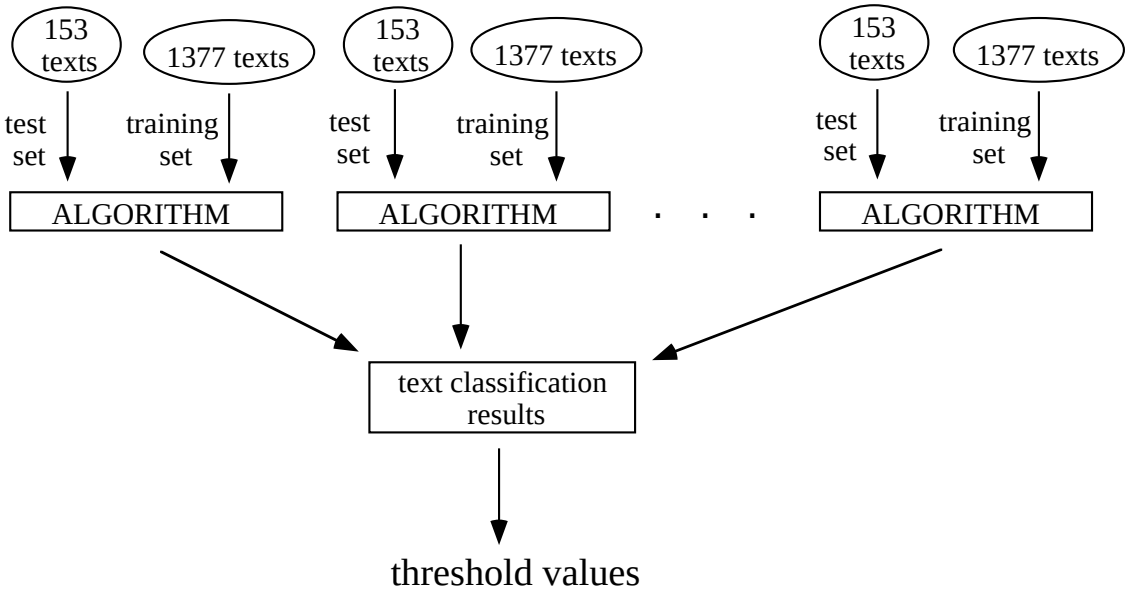


Figure 4.10: Threshold experiments

the results of all the folds for each set of thresholds values. For example, we would sum all of the classification results for the relevancy signatures algorithm with threshold values: {.85, 10}. This sum reflects the performance of the algorithm on all 1530 texts using threshold values {.85, 10}. Finally, we determine which threshold settings achieved the best performance under each metric and return these seven sets of threshold values.

It is important to remember that we derive a different set of threshold values for each fold! Figure 4.9 shows that the training set for each fold is also input to the threshold experiments. The resulting threshold settings are then applied only to the test set for the same fold. Therefore, each fold may use a different set of "best" threshold values.

The motivation for this design is that it demonstrates how a user might go about finding good threshold settings in a real-world scenario. Given a training set (which is necessary to generate statistical data anyway), a user can use the same training set to empirically derive a good set of threshold values. Assuming that the training texts are representative of new texts, the results obtained in the threshold experiments predict the level of performance that will be achieved on novel texts.

Finally, we evaluate the test sets for each fold using the "best" threshold settings for that fold. Each fold therefore yields seven sets of results, one for each metric. The results of the folds are combined to yield seven final data points that reflect the performance of the algorithm on the entire corpus. In the following sections, we use this procedure to evaluate the text classification algorithms in three different domains: the MUC-4 terrorism domain, a joint ventures domain, and a microelectronics domain.

4.6.2 The Terrorism Domain

As we noted earlier, the experiments presented so far are based on 200 blind texts from the MUC-4 corpus. Although we saw substantial performance differences for the different algorithms on these texts, we need to evaluate the algorithms more thoroughly on a larger collection of texts. Therefore we evaluated the algorithms on the basis of the entire MUC-4 corpus (1700 texts) using

the 10-fold cross-validation design and the procedure to identify the best threshold values described in the previous section. The results of this experiment are shown in Figure 4.11.[16]

It is reassuring to see that the results in Figure 4.11 are consistent with the results obtained in the previous experiments. In general, relevancy signatures achieve better precision than the relevant words, often at comparable or higher recall levels (relevancy signatures achieve better results under 6 of the 7 metrics). However, both augmented relevancy signatures and the case-based algorithm perform better than the other two algorithms. In general, augmented relevancy signatures and the case-based algorithm produce very similar results. Augmented relevancy signatures obtain the best precision, which is not surprising since it has the most rigid criteria for relevance. For the most part, the case-based algorithm achieves slightly better recall than augmented relevancy signatures with similar levels of precision (some higher, some lower).

Remember that the threshold settings were derived empirically but they were not guaranteed to produce the "best" results on novel texts. Therefore each of the algorithms may be capable of achieving better performance than we see here. If a test set has significantly different properties than the training set then the empirically derived threshold values may do poorly whereas different threshold values would have done better. By evaluating the algorithms on the entire corpus of 1700 texts, we hoped to minimize these effects. Nevertheless, the consistency of one algorithm to outperform another is compelling, as is the consistency of these results with those reported in the earlier experiments.

Finally, the performance levels shown in Figure 4.11 are not quite as high as those attained on TST3 and TST4. For example, we do not achieve 100% precision under any metric and, in general, the precision levels are lower. One explanation for this is that there is some noise associated with the answer keys in the MUC-4 corpus. As we mentioned in Section 2.2, most of the answer keys were generated by the MUC-3 and MUC-4 participants. We know from experience that some parts of the corpus were more reliably encoded than others. In contrast, the answer keys associated with the TST3 and TST4 texts were generated by the conference organizers. Because these answer keys were used for the final evaluation, extra care was taken to ensure that they were encoded as carefully as possible. In general, noisy answer keys impact both the training data and the test results. But since our algorithms are based on statistical patterns, they should be tolerant of some noise in the training data. Noise in the test set, however, has direct impact on the final results.

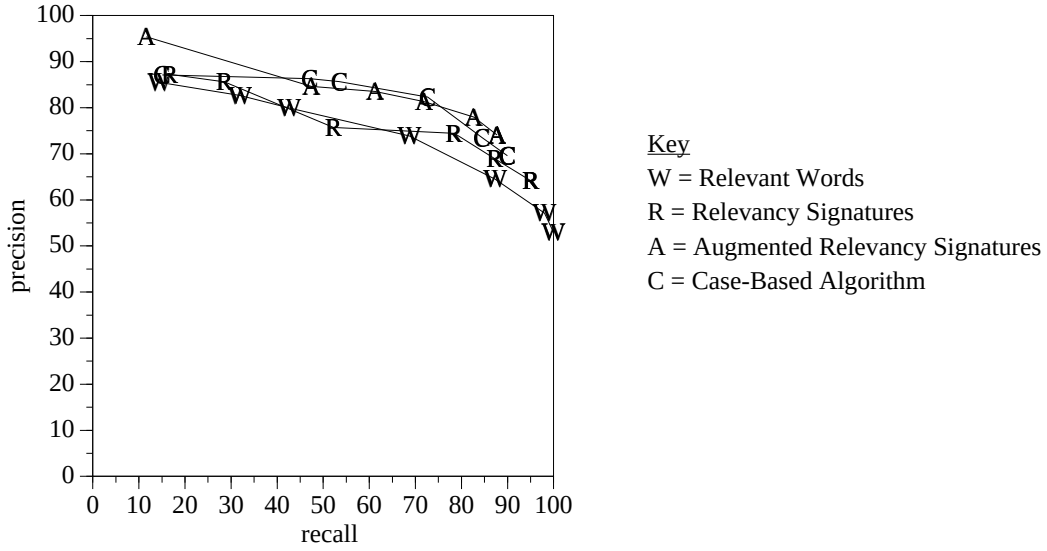
4.6.3 The Joint Ventures Domain

To gain a better understanding of the general strengths and weaknesses of the text classification algorithms, we evaluated them on two additional domains: the MUC-5 joint ventures domain and the MUC-5 microelectronics domain. In this section, we describe the steps that were necessary to bring up the text classification algorithms in the new domains and present experimental results.

4.6.3.1 Generating a Training Corpus

The text classification algorithms all rely on a training corpus of relevant and irrelevant texts. The MUC-5 participants were provided with a development corpus for the joint ventures domain, but the corpus was highly skewed toward relevant texts: 92% of the texts in the JV corpus were relevant. Therefore the MUC-5 corpus was not appropriate because the text classification algorithms require a training corpus that contains roughly an equal mix of relevant and irrelevant texts. A balanced corpus is necessary because the statistics used by the algorithms identify associations between extracted information and relevance. If a corpus contains mostly relevant

[16]Some of the algorithms generated fewer than seven data points because multiple metrics produced the same results.



Algorithm	Recall	F(2)	F(1)	F(.5)	F(.3)	F(.2)	Precision
Rel Words	100 53	98 57	87 65	69 74	43 80	32 83	15 86
Rel Sigs	95 64	95 64	87 69	78 74	52 76	29 86	17 87
Aug Rel Sigs	88 74	88 74	83 78	72 81	61 84	47 85	12 95
CBR	90 70	90 70	84 74	73 82	53 86	47 86	15 87

Figure 4.11: Graph of cross-validation results

texts, then virtually every phrase or context will be highly correlated with relevant texts. If a corpus contains mostly irrelevant texts then few, if any, phrases or contexts will be highly correlated with relevant texts. The ideal training corpus should be relatively balanced between relevant and irrelevant texts.

So we needed to augment the corpus with irrelevant texts to balance it. To complicate matters, the irrelevant texts must be "near-miss" texts; that is, they should contain the same "types" of information as the relevant texts. For example, the irrelevant texts in the MUC-4 corpus typically describe military incidents, civilian crimes, or terrorist activities in general. These stories frequently refer to violent crimes, perpetrators, victims, targets, and weapons, so they activate many of the same concept nodes as the relevant texts. This is important because the probability estimates need both positive and negative examples to identify reliable phrases and contexts.

To generate appropriate near-miss texts, we acquired additional texts from the corpus used in the Tipster Text Detection evaluation [Tipster Proceedings, 1993, Harman, 1992a]. The Tipster Text Detection evaluation was a competitive performance evaluation of information retrieval systems. *Text Routing* was one of the Tipster detection tasks; the problem is: given a topic description, retrieve all documents from a corpus that are relevant to the topic. One of the topics given to the detection sites was about joint venture activities. Although the JV detection topic was not exactly the same as the MUC-5 domain, it was close enough to use as a starting point.[17]

[17]Among other things, the JV detection topic required that one of the entities involved in the joint venture must be Japanese.

So we used texts from the Tipster detection corpus to augment the MUC-5 corpus. The Tipster detection texts came from three different news sources: the Wall Street Journal, the AP news wire service, and Ziff Publications.

First, we collected the texts that were retrieved by at least one of the detection sites in response to the JV topic description. The detection texts had been manually reviewed by human experts and judged to be either relevant to the JV topic or irrelevant. Since we needed only *irrelevant* texts, we gathered the texts that were retrieved by a detection system but were subsequently judged to be *irrelevant* to by one of the reviewers. These texts represent false hits by the detection systems. This set of documents has two important properties:

1. These texts are irrelevant to the JV detection topic.
2. These texts were mistakenly classified as relevant by a detection system.

The first property is necessary because we need irrelevant texts to augment the MUC-5 corpus. The second property is important because we want "near-miss" texts. There are many reasons why a detection system might retrieve an irrelevant document, so there is no guarantee that these texts actually contain the same types of information as the relevant texts. However, most of the texts were retrieved because they contained words that were relevant to the domain. Consequently, we assume that many of them contain information that is related to the domain. This assumption will not always be true but the advantage of being able to acquire these texts automatically outweighs the potential disadvantage of having some poor quality near-misses.

At this point, we had 928 texts that were irrelevant to the JV detection topic. But that doesn't necessarily mean that they were irrelevant to the MUC-5 domain. For example, the JV detection topic required that one of the partners must be Japanese. A text that describes a joint venture between two American companies is irrelevant to the JV detection topic but relevant to the MUC-5 domain. To be sure that the newly acquired texts were truly irrelevant to the MUC-5 domain, we manually reviewed the 928 texts and found that 481 of the texts were irrelevant to MUC-5 domain.

Since the MUC-5 corpus contained 923 relevant texts, which is more than we needed, we randomly selected a subset of them to use for the experiments. To create a JV training corpus with roughly a 50-50 combination of relevant and irrelevant texts, we combined the 481 irrelevant texts from the Tipster detection corpus, the 76 irrelevant texts from the MUC-5 corpus, and 643 randomly selected relevant texts from the MUC-5 corpus. The final JV training corpus consisted of 1200 texts of which 54% were relevant.

4.6.3.2 Generating a Semantic Feature Dictionary

The augmented relevancy signatures algorithm and the case-based text classification algorithm both depend on semantic features associated with words in the dictionary. The UMass/MUC-5 system for the joint ventures domain did not use a semantic feature dictionary.[18] Therefore we had to construct our own semantic feature dictionary for the joint ventures domain.

Although the task of creating a semantic feature dictionary might initially seem like a time-consuming effort, it can be done rather quickly by taking advantage of the corpus used by AutoSlog. We do not claim that a complete semantic feature dictionary for the English language can be created quickly! But a core dictionary for the most common words in the domain can be acquired with a small amount of effort.

---

[18]The UMass/MUC-5 system did use a system called Maytag [Cardie, 1993] that assigned semantic features to words dynamically. In theory, Maytag could have been used with the text classification algorithms but we chose not to use it because of time constraints.

As we described in Section 3.5.2, AutoSlog used the MUC-5 training corpus to automatically create a concept node dictionary for the joint ventures domain. The key templates contain the noun phrases that should have been extracted from each text. Each noun phrase has a slot type that specifies what type of information it is; for example, the JV templates contain slots for partners, child companies, people, products, revenue amounts, etc. We can use the information in the templates to reverse-engineer a coarse semantic feature dictionary.[19] To generate a simple first-pass dictionary, we extracted the head noun of every noun phrase in a template and assigned it a default feature associated with its slot. For example, head nouns associated with partners were automatically assigned the semantic feature *company*, head nouns associated with products were assigned the semantic feature *material-good*, etc. This process was fully automated so the only human effort was to specify a set of mappings from template slot types to general semantic features.

The resulting semantic feature dictionary is not perfect. For one, the same head noun can appear in different noun phrases from different template slots. Therefore some words are assigned more than one semantic feature. The quality of the semantic features may also be too general for some slots; finer-grained distinctions may be necessary to produce good performance. And head nouns are not always sufficient to represent the semantic meaning of a noun phrase, so some of the dictionary entries are not appropriate.

To generate a more dependable semantic feature dictionary, a second-pass through the dictionary is necessary. We created a simple interface that allows a user to quickly scan each entry and either keep the original semantic feature(s) or change it. This process allows a user to fix incorrect features or replace a general feature with a more specific one. To make things simple, we allowed only a single semantic feature per word.

The first-pass dictionary that was created automatically contained 2726 words. Then we manually revised the dictionary entries. For example, words that were originally assigned the *product* feature were separated into several different types: *animal*, *chemical*, *finance*, *food*, *material-good*, *natural-resource*, *research*, *service*, *vehicle*. In general, we tried to create categories that were abstract enough to include more than a few words but specific enough so that words in the same category shared a similar purpose, function, or property. The categories were motivated entirely by the words in the corpus. After the dictionary was revised, we scanned the dictionary a third time to make sure that the entries were relatively consistent with one another. At this point, the dictionary was quite stable so only a small number of entries needed to be changed.

To summarize, we created a semantic feature dictionary by reverse-engineering the key templates for the domain. A first-pass dictionary was created automatically by assigning the head noun of each slot filler with a default feature corresponding to the slot type. The first-pass dictionary was then revised manually by scanning the dictionary entries and changing or adding more specific semantic features as needed. The process of reviewing dictionary entries is very fast because most of the default features are adequate. For example, company names were assigned the default feature *company* and people names were assigned the default feature *person* and very few of these needed to be changed.

The final list of semantic features for the joint ventures domain is shown in Appendix 7.3, along with the number of words assigned to each category and examples.[20] Finally, the semantic feature

---

[19]Although we used key templates in this experiment, this process would work equally well with annotated texts (see Section 5.4.3). Given an annotated corpus, one could use the semantic tags associated with the marked noun phrases.

[20]Most of the features are self-explanatory but a few of them need clarification. The *entity* feature is a superclass for the company and person categories. A word was assigned to the *entity* class if it was a generic reference to a group (e.g., association and consortium) or if it was a

dictionary was used to evaluate the augmented relevancy signature algorithm and the case-based text classification algorithm in the joint ventures domain. For each noun phrase extracted by a concept node, the semantic feature associated with the head noun was used to replace the noun phrase. The semantic class of the head noun of a noun phrase taken out of context is not always the same as the semantic class of the noun phrase as a whole. Determining the semantic class associated with a noun phrase is a difficult task (Cardie [Cardie, 1993] describes a system that attempts to do this.), but, for our purposes, we were willing to assume that using the semantic class of the head nouns was a reasonable approximation.

4.6.3.3 Experimental Results

We evaluated the text classification algorithms in the joint ventures domain using the same cross-validation design described in Section 4.6.2. That is, we randomly partitioned the JV corpus of 1200 texts into 10 folds, where each fold contained a test set of 120 texts and a training set of 1080 texts. For each fold, we used the experimental procedure described in Section 4.6.1 to empirically derive good threshold settings for seven different metrics.

The results are shown in Figure 4.12. The relevant words algorithm did not perform nearly as well as the others. Looking at the "relevant words" that were chosen provides some insight. We examined the performance of the algorithm on one specific fold of cross-validation and extracted the relevant words that were chosen under the **Precision** metric with threshold values (90 50). These threshold values are very high so the words that pass the thresholds are the ones most highly correlated with relevant texts in the corpus. Eight words were judged to be "relevant" using these threshold values: consumption, Hungarian, Indonesian, Malaysia, Malaysian, pct, Perot, Pt. These words fall into three categories:

1. Pct, Pt
2. Hungarian, Indonesian, Malaysia, Malaysian
3. Consumption, Perot

The words in category (1) make sense. Pct is an abbreviation for "percent" used by CIRCUS and percentages are an important concept in joint ventures. PT is the Indonesian equivalent of a company designator, similar to Inc. or Corp. Although PT is not specific to joint ventures, it is a symbol of companies which are an important component of joint ventures activities.

On the other hand, the words in category (2) are adjectives or nouns describing countries. The fact that these words made it onto the list illustrates an important aspect of the text classification task. The task is not necessarily to find concepts that are indicative of a domain, but to find concepts that are useful for *discriminating* relevant texts from irrelevant texts. In the MUC-5 corpus, almost all of the texts that contain the words "Hungarian", "Indonesian", "Malaysia", or "Malaysian" discuss joint venture activities. A purely statistical algorithm, like the relevant words

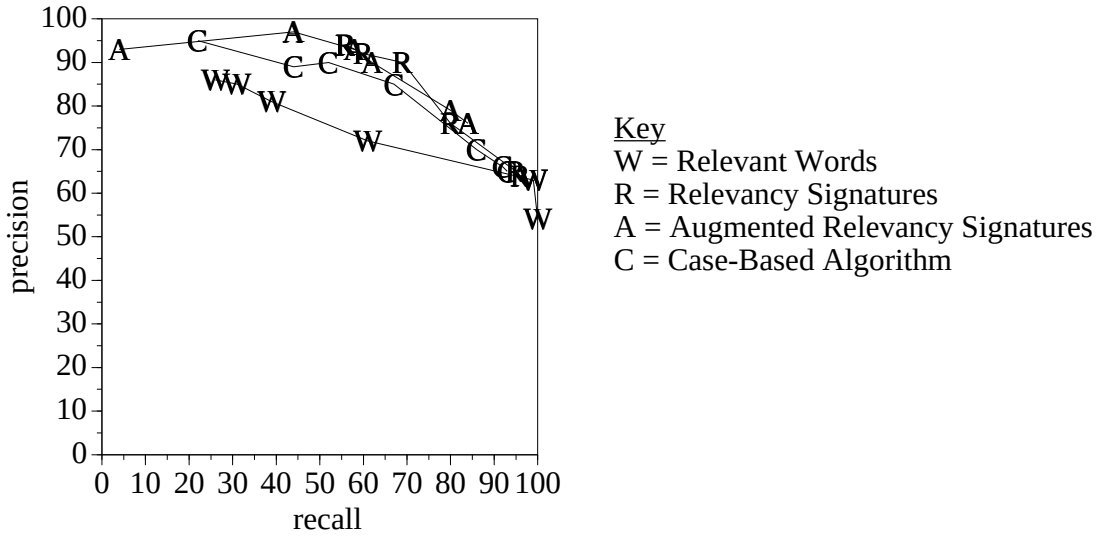
---

proper-name that could refer to a person or a company (e.g., Baxter). Since many companies are named after people, it is often difficult to distinguish company names and human names out of context. Since both companies and individuals can be partners in a joint venture, a superclass seemed warranted. The *thing* feature is a non-feature that represents everything else; a word was assigned the feature *thing* if it did not fall into any of the existing classes and did not have any relevance to joint venture activities. This category was often used for abstract words such as "activity" and "combination", as well as seemingly relevant words like "joint-venture". The word "joint-venture" itself is an abstract notion that is important for triggering case frames but does not represent an object.

algorithm, cannot distinguish between words that are meaningful indicators of a domain from words that coincidentally happen to be highly correlated with the domain in a training corpus.

In fact, these words may be useful for classifying future texts if the future texts follow the same trend as the training corpus. But these words are risky because their relevance is an artifact of the corpus and not the domain. One of the advantages of the other text classification algorithms is that they are based on an underlying information extraction system that was developed for a specific domain. Therefore the features used for classification are semantically related to the domain so correlations with relevant texts are less likely to be artifacts of coincidence or circumstance.

The relevancy signatures algorithm performed extremely well on the JV domain, achieving scores of 56% recall with 94% precision and 69% recall with 90% precision. Augmented relevancy signatures achieved nearly the same level of performance, reaching 58% recall with 93% precision and 44% recall with 97% precision.



Algorithm	Recall	F(2)	F(1)	F(.5)	F(.3)	F(.2)	Precision
Rel Words	100 54	99 63	99 63	61 72	39 81	31 85	26 86
Rel Sigs	96 64	95 65	80 76	69 90	60 92	56 94	56 94
Aug Rel Sigs	84 76	84 76	80 79	62 90	58 93	44 97	4 93
CBR	93 65	92 66	86 70	67 85	52 90	44 89	22 95

Figure 4.12: Graph of cross-validation results for joint ventures

The augmented relevancy signatures data point for the **Precision** metric (4% recall with 93% precision) is misleading, but it is useful to understand how it came to be. Table 4.3 shows the results for the augmented relevancy signatures algorithm under the **Precision** metric under individual folds. The second column shows the threshold values chosen by the threshold derivation procedure, and the third and fourth columns show the number of relevant texts correctly classified as relevant and the total number of texts classified as relevant.

Fold 9, which classified 24 texts as relevant, produced dramatically different results than the other folds. The reason is that Fold 9 used a relevancy threshold of 85% but the other folds used a relevancy threshold of 95%. Given the same frequency cutoffs, the 95% threshold produces fewer

relevancy signatures than the 85% threshold. But Fold 9 used a frequency threshold of 15 (i.e., the corpus must contain at least 15 occurrences of the signature) and the other folds used a frequency threshold of 0 (i.e., the corpus must contain at least 0 occurrences of the signature). As a result, Fold 1 identified 419 relevancy signatures but Fold 9 identified only 21 relevancy signatures! A frequency threshold of 15 substantially reduces the number of signatures that are eligible to become relevancy signatures.

Given the substantially smaller number of relevancy signatures used by Fold 9, it is surprising that it found more relevant texts than the other folds. Upon further inspection, we found that the signature <venture, $jv-entity-subject-verb-and-dobj-venture$> was in the list of relevancy signatures for Fold 9, but none of the other folds. This signature is activated whenever the noun "venture" is preceded by a verb, so it is represents a very general pattern that is extremely common in the joint ventures domain. In fact, *every* text classified as relevant by Fold 9 relied on this signature. When we removed this one signature for the list of relevancy signatures for Fold 9, *none* of the texts were classified as relevant. The presence of this single relevancy signature had a major impact on performance.

Table 4.3: Augmented relevancy signatures results for JV under **Precision**

Fold	**Thresholds**	**#correctly classified**	**#total classified**
Fold 1	(95, 0, 95, 5)	0	0
Fold 2	(95, 0, 90, 10)	1	1
Fold 3	(95, 0, 95, 0)	0	0
Fold 4	(95, 0, 95, 5)	0	0
Fold 5	(95, 0, 95, 10)	0	0
Fold 6	(95, 0, 95, 0)	0	0
Fold 7	(95, 0, 95, 0)	0	0
Fold 8	(95, 0, 95, 0)	2	2
Fold 9	(85, 15, 95, 0)	22	24
Fold 10	(95, 0, 95, 0)	1	1

The story behind this data point illustrates some of the weaknesses of this evaluation scheme. First, it demonstrates that the procedure for empirically deriving threshold values does not always produce consistent results. The threshold values are chosen based upon the performance of the algorithm on a training corpus. If the training corpus has properties that are atypical of the test corpus, then the values recommended by the procedure will not necessarily be the ones that perform best on novel texts. Because of the empirical nature of the procedure, the threshold values should not be interpreted as optimal values but only as "suggested" threshold values that are likely to produce the desired results.

Second, the cross-validation design is strange because each fold is evaluated separately on a different training set. Even though the corpus is randomly partitioned, it is possible that one or more of the folds contains a subset of the corpus that is skewed in one way or another. If so, then one or more of the folds can produce results that are substantially weaker (or stronger) than the other folds. If the skewed effect is large enough then the cumulative results are affected. However, if the skewed effect is small then the difference will be swamped out by the other folds and the cumulative results are not significantly affected. The lesson is that the cross-validation design is a powerful technique for exploiting a relatively small corpus for both testing and training purposes, but the results should be taken with a grain of salt and interpreted only as ballpark estimates of the general performance of an algorithm.

It is not a coincidence that the results for the relevancy signatures algorithm and the augmented relevancy signatures algorithm in the joint ventures domain are similar. As Section 4.4.2 explained, the augmented relevancy signatures algorithm classifies a text as relevant only if it contains both a relevancy signature and a reliable slot triple. However, only a small number of signatures were highly correlated with relevance in the JV corpus. Furthermore, almost all of relevancy signatures represented concept nodes to extract entities. Therefore only the fillers extracted by these concept nodes are eligible to become reliable slot triples. A slot triple is created by replacing the noun phrase extracted by a concept node with the semantic feature associated with its head noun. But a joint venture entity must have one of three semantic feature types (company, government, person), so only a small set of slot triples are considered as candidates. For example, Table 4.4 shows the patterns of the 21 relevancy signatures used by Fold 9.

Table 4.4: JV relevancy signature patterns used by Fold 9

<entity> owned PERCENT-OBJECT
PERCENT-OBJECT by <entity>
plans to build <product-service>
<entity> plans to build
capital of <ownership-total-capitalization>
was capitalized at <ownership-total-capitalization>
<entity> was capitalized
<entity> agreed to form
<ownership-percent> was owned
<entity> build plant
set up with <entity>
<entity> <verb> venture
<entity> set up venture
<entity> formed venture
<entity> form venture
<entity> established venture
<entity> establish venture
<entity> is venture
venture with <entity>
venture of <entity>
venture between <entity>

Seventeen of these 21 signatures are patterns to extract entities. Since the majority of the signatures extract entities, this implies that the majority of eligible slot triples correspond to semantic features associated with entities. Therefore a small subset of the slot triples are accessed much more frequently than the others. Furthermore, joint ventures entities should be companies, governments, or people, so a small subset of the semantic features are accessed a lot more than others.

Finally, the case-based algorithm produced good results for the joint ventures domain but did not perform quite as well as the relevancy signatures and augmented relevancy signatures. After manually inspecting the output, we concluded that the main reason is that key phrases are crucially important in the joint ventures domain.

For example, the case-based algorithm incorrectly classified the following sentence as relevant:

Of its capital of 10 million Singapore dollars, 50 percent was put up by the Japanese firm.

This sentence was classified as relevant because it generated the following relevancy index:

(<put, $jv-own_percent-subject-passive-verb-put-up$>,
(entities, THING),
(total-caps percents entities))

This index represents the fact that the sentence contains the phrase "was put up", that an entity of type THING was extracted ("the Japanese firm"), and that the sentence contained information about total capitalization, percentages, and entities ("10 million Singapore dollars", "50 percent", and "the Japanese firm"). The corpus contained seven cases that shared exactly the same information and 100% of these cases came from relevant texts.

This sentence contains a lot of information that is common to joint ventures. However, it is missing something crucial: a reference to a joint venture. The information in this sentence is also common in many other types of texts. In the joint ventures domain, the presence of a key phrase referring to a joint venture is critical. It is possible to describe a joint venture solely by context but, in practice, almost every text describing a joint venture uses a key phrase to identify the act as a joint venture.

Another factor that may have contributed to the slightly poorer performance of the case-based algorithm is that it relied on relatively few cases to make relevance judgements. In most cases, the case-based algorithm used a frequency threshold of 5. A larger case base might have provided more examples and therefore more reliable statistics.

Despite these problems, the case-based algorithm performed well and was able to achieve high precision with reasonable levels of recall. Figure 4.12 shows that the case-based text classification algorithm achieved 95% precision with 22% recall and 90% precision with 52% recall.

4.6.3.4 Comparing Algorithms Using Standard Deviations

So far, we have evaluated the text classification algorithms in two different domains and shown that the IE-based algorithms performed better than the relevant words algorithm. Ideally, it would be nice to know whether the differences between the algorithms are statistically significant. However, the cross-validation design that we used to evaluate the algorithms is complicated by the fact that each fold of the cross-validation was tested with a different set of seven threshold values. Therefore it does not make sense to do a point-by-point comparison of the algorithms.

To get some idea of whether the algorithms behave substantially differently from one another, we compared the standard deviations of the precision values over all of the cross-validation folds for the relevant words algorithm and the relevancy signatures algorithm in the joint ventures domain. If the standard deviation graphs for different algorithms do not overlap, then the performance results achieved by the algorithms fall into different ranges that are distinct. Although we only analyzed the results of two of the algorithms for one of the domains, this procedure could be used to compare any pair of algorithms in any domain. The procedure is as follows:

1. Collect all 70 data points produced by the cross-validation experiment.[21]
2. Partition the data points by their recall values into ten groups. For example, put the data points with recall values between 90-100% in partition #1, put the data points with recall values between 80-90% in partition #2, etc.
3. For each partition, compute the average precision value and the standard deviation of the precision values of all the data points.

[21]Each of the ten folds produced seven data points, one for each of the seven metrics.

Using this procedure, we generated three data points for each partition by pairing the midpoint of the recall range with the average precision value, the average precision value plus the standard deviation, and the average precision value minus the standard deviation. Tables 4.5 and 4.6 show the values of each partition for each algorithm. The original 70 data points were not evenly distributed across the different recall ranges, so some partitions had fewer data points than others. For example, the relevant words algorithm produced no data points with recall values between 70-90%, and only one data point with a recall value between 0-10% (so the standard deviation was zero). The gap in recall values reflects the difficulty that the algorithm had with achieving high precision. The relevant words algorithm can obtain very high recall (90-100%) but only with an average precision of 60.70%. To get higher precision the algorithm must drastically increase the threshold values which substantially reduces recall.

Table 4.5: Standard Deviation Results for Relevant Words in JV

Recall Midpoint	Avg Precision	Avg Precision + Standard Dev.	Avg Precision - Standard Dev.
5.00	62.50	62.50	62.50
15.00	85.19	95.66	74.71
25.00	83.72	90.40	77.04
35.00	86.92	93.18	80.66
45.00	82.00	89.32	74.68
55.00	72.74	78.06	67.43
65.00	73.33	76.36	70.31
75.00	–	–	–
85.00	–	–	–
95.00	60.70	67.91	53.49

The relevancy signatures algorithm, on the other hand, produced no data points with a recall value under 40%. Relevancy signatures are a richer representation than individual words so the relevancy signatures algorithm does not have to increase the threshold values as dramatically as the word-based algorithm. As a result, relevancy signatures can achieve an average precision of 75-90% at high recall levels of 70-90%.

Figure 4.13 shows the curves generated by the data points in Tables 4.5 and 4.6. The graph shows three curves for each algorithm: the middle curve represents the average precision for a fixed recall range, and the two curves around it represent the standard deviation of the precision values for that recall range. Figure 4.13 shows that the standard deviation curves for the relevancy signatures algorithm do not overlap with the standard deviation curves for the relevant words algorithm in the recall ranges below 80%. This graph implies that the results achieved by the algorithms fall into distinct ranges. We conclude that the relevancy signatures algorithm performed distinctly better than the relevant words algorithm in the joint ventures domain for recall values below 80%.

4.6.4 The Microelectronics Domain

We also evaluated the relevant words algorithm and the relevancy signatures algorithm in the MUC-5 microelectronics domain. However, we did not evaluate the augmented relevancy signatures algorithm or the case-based algorithm in the microelectronics domain because those

Table 4.6: Standard Deviation Results for Relevancy Signatures in JV

Recall Midpoint	Avg Precision	Avg Precision + Standard Dev.	Avg Precision - Standard Dev.
5.00	–	–	–
15.00	–	–	–
25.00	–	–	–
35.00	–	–	–
45.00	94.17	96.67	91.67
55.00	94.44	97.20	91.67
65.00	91.37	95.82	86.92
75.00	88.48	93.54	83.42
85.00	74.78	79.77	69.80
95.00	64.82	71.08	58.57

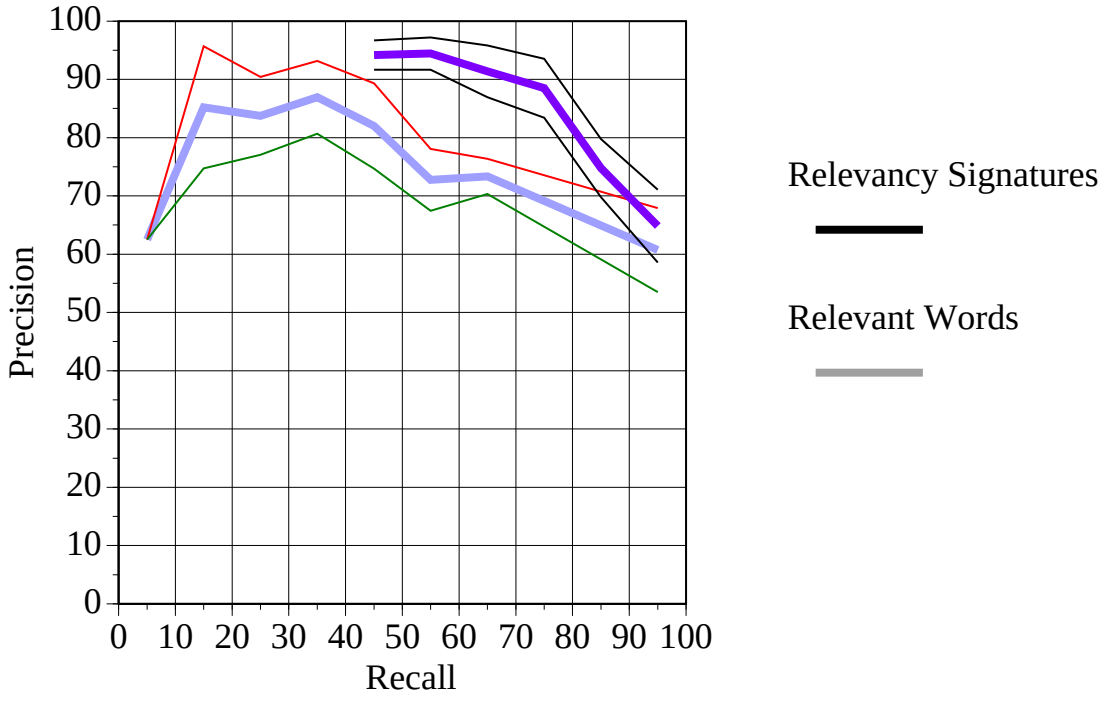
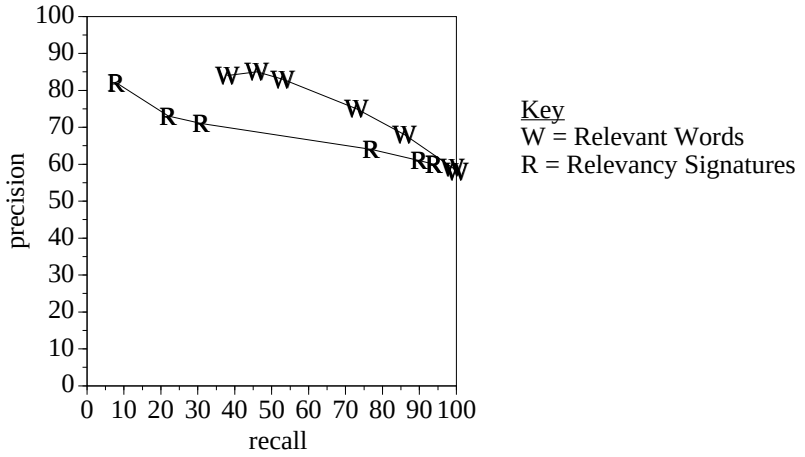


Figure 4.13: Standard deviation curves in the joint ventures domain.

algorithms rely on semantic features. We did not feel that we understood the microelectronics domain well enough to assign meaningful semantic features to microelectronics concepts.

Figure 4.14 shows the results for the relevant words algorithm and the relevancy signatures algorithm on the MUC-5 microelectronics corpus using the same cross-validation design described previously. In this domain, relevancy signatures were not as effective as with terrorism and joint ventures. The highest precision achieved by relevancy signatures was 82% precision with only 8% recall. In contrast, relevant words did much better. The relevant words algorithm achieved 85% precision with 46% recall.



Algorithm	Recall	F(2)	F(1)	F(.5)	F(.3)	F(.2)	Precision
Rel Words	100 58	99 59	86 68	73 75	53 83	46 85	38 84
Rel Sigs	94 60	94 60	90 61	77 64	31 71	22 73	8 82

Figure 4.14: Graph of cross-validation results for microelectronics

The word-based algorithm did not achieve extremely high precision (i.e., $\geq$ 90%) but it performed substantially better than the relevancy signatures. The reason is best understood by showing the words and signatures that were used by the algorithms. Using Fold 1 as an example, the relevant words algorithm chose threshold values (95 30) which produced 23 keywords:

Table 4.7: ME relevant words used by Fold 1 under **Precision**

-MEGABIT	CYPRESS	LAM	NOVELLUS
-NS	FLASH	MCM	NS
-V	GCA	MCMS	STEPPERS
AT&T	GENUS	MOTOROLA	TSOP
BICMOS	HAMPSHIRE	MULTICHIP	ULTRATECH
BIPOLAR	I-LINE	NIKON	

The relevancy signatures algorithm, however, used threshold values of (95 16) which produced only a single relevancy signature: <stepper, $me-entity-subject-verb-and-dobj-stepper$>. This

signature is activated whenever the word "stepper" is preceded by any verb. The fact that only a single signature was used for classification explains why the relevancy signatures algorithm got only 8% recall.

Under the F(.2) metric, the relevant words algorithm used the same threshold values but the relevancy signatures algorithm used threshold values (90 8). These lower threshold settings produced 25 signatures, however most of the signatures did not represent reliable microelectronics phrases. For example, the relevancy signatures included general expressions such as "<X> agreed", "develop <X>", "manufactures <X>", and "<X> was received". They also included phrases that contained technical words, such as MCMS, DRAMS, and STEPPERS. But overall precision suffered from the presence of the general expressions as well as the technical phrases.

To summarize, the relevant words algorithm performed better than the relevancy signatures algorithm in the microelectronics domain because the signatures did not represent most of the technical words that are relevant to the domain. The relevancy signatures algorithm could identify only a few reliable signatures with strong relevancy correlations and, as a result, achieved good precision only with low recall. To get higher recall, the algorithm was forced to lower the threshold values which resulted in less reliable phrases and lower precision. The relevant words algorithm, however, was not constrained by the concept node dictionary and was able to identify many words that were strongly associated with the domain. In Chapter 5, we discuss general properties of domains and tasks, including distinctions between technical and event-driven domains, that determine the effectiveness of the classification algorithms.

4.7 Additional Experiments with Multiple Relevancy Signatures

Most information retrieval systems judge the relevance of a text by looking at multiple indexing terms. In general, the relevance of a text is determined by looking at the number of key words or phrases contained in the text and the importance (usually, weighting) of the terms. Texts that contain multiple sources of evidence for a domain are usually ranked more highly than texts that contain only a single source of evidence. Using multiple indexing terms is one way to capture global context and represent the topic of a text.

The text classification algorithms presented in this chapter determine the relevance of a text based solely on a single piece of evidence. For the relevancy signatures algorithm, the presence of single relevancy signature is enough to classify a text as relevant. Texts that contain multiple relevancy signatures are not considered to be any more relevant than texts that contain only one.

Similarly, the augmented relevancy signatures algorithm classifies a text as relevant if it contains a single augmented relevancy signature. An augmented relevancy signature consists of two items: a relevancy signature and a relevant slot triple. However, both items must have originated from a single concept node so the information represented by an augmented relevancy signature is a single piece of information (e.g., the fact that a government official was assassinated). We view an augmented relevancy signature as a single indexing term, although with a richer representation.

The case-based text classification algorithm also classifies a text as relevant if it finds a single case in the text that is judged relevant. Cases are a much richer representation than relevancy signature because they contain information that may span multiple clauses in a sentence. However, a case always contains information from the same sentence. Therefore we also view a case as a single, albeit very rich, indexing term. In summary, none of the text classification algorithms that we have described account for information that is distributed throughout a text.

The strategy of relying on a single indexing term was deliberate. Many of the terrorism texts in the MUC-4 corpus contain sentences that we refer to as *fleeting references*. A fleeting reference is a brief mention of an event that appears in the larger context of a different event or topic. For example, a news article reporting on a bombing incident may mention somewhere at the end of

the article that the bombing was in retaliation for the assassination of a party leader. Or a text reporting on the political situation in San Salvador might mention somewhere in the middle of the second page that a recent kidnapping changed the nature of upcoming elections. In particular, many texts in the MUC-4 corpus contain a fleeting reference to the assassination of the jesuits priests in El Salvador in the context of other news events.

A classification algorithm that judges the relevancy of a text by looking for multiple pieces of evidence will be unlikely to recognize texts with fleeting references as relevant texts. Typically, only a single sentence contains relevant information and the rest of the text (which could be many pages) contains information about completely different subjects. There is no right or wrong answer about which approach is better. It depends entirely on the task. If a user is interested in finding all texts that *focus* on a specific topic, then it is best to use an algorithm that considers the context of the entire article. However, if a user is interested in finding all texts that *mention* a specific topic, then it is best to use an algorithm that requires only a single reference to the topic. In practice, many texts discuss several different topics and it is not realistic to assume that most texts focus on a single topic. Even news wire articles, which are generally short and more direct than other texts, are surprisingly diverse and often contain paragraphs about many different topics.

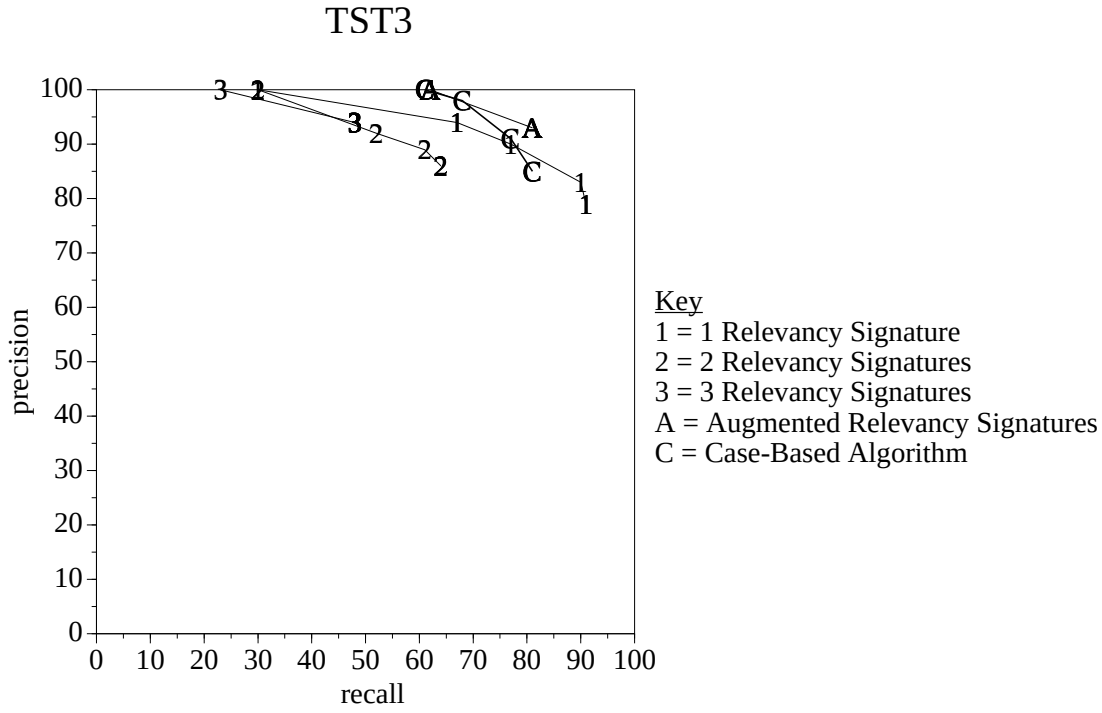
For the MUC-4 terrorism domain, a text was considered to be relevant if it contained any reference to a specific terrorist incident. Similarly, the MUC-5 joint ventures and microelectronics domains considered a text to be relevant if it contained any reference to a relevant joint venture or microelectronics activity. Because these three domains were all defined in this manner, we purposely designed the algorithms for these sorts of tasks. One can easily imagine similar topics that might be of interest for real applications, such as texts that contain any reference to business activities of a specific company, texts that contain any reference to a certain stock, or texts that contain any reference to the actions of a particular government.

Nevertheless, we ran a set of experiments to see whether the performance of the relevancy signatures algorithm would improve if it used multiple signatures. We tested two modified versions of the algorithm which worked exactly like the original except that a text was classified as relevant only if contained at least 2 or 3 relevancy signatures, respectively. We ran the algorithms in the terrorism domain using the 1500 texts in the MUC-4 development corpus for training and TST3 and TST4 for testing. Figures 4.15 and 4.16 show the results of this experiment. For reference, these figures include the results presented previously for the original relevancy signatures algorithm, the augmented relevancy signatures algorithm, and the case-based algorithm.

We will refer to the original relevancy signatures algorithm as Rel-Sigs-1, the version requiring two relevancy signatures as Rel-Sigs-2, and the version requiring three relevancy signatures as Rel-Sigs-3. On TST3, Rel-Sigs-2 and Rel-Sigs-3 achieved better precision than Rel-Sigs-1 at the high recall end of the spectrum but with substantially lower recall. The highest recall attained by Rel-Sigs-2 was 64% (with 86% precision) and the highest recall reached by Rel-Sigs-3 was 48% (with 94% precision). Note that the original relevancy signatures algorithm obtained 77% recall with 90% precision and 67% recall with 94% precision so there wasn't any overal improvement from using the extra signatures. The precision gained by requiring multiple signatures was at the expense of significant recall.

However, the results for TST4 were slightly different. At the high recall end, the results are similar; precision improves but with substantial recall loss. At the high precision end, though, Rel-Sigs-2 was able to get 58% recall with 94% precision and 31% recall with 100% precision. Similarly, Rel-Sigs-3 was able to get 55% recall with 91% precision and 20% recall with 100% precision. Both of these algorithms performed better than Rel-Sigs-1 at the high precision end.

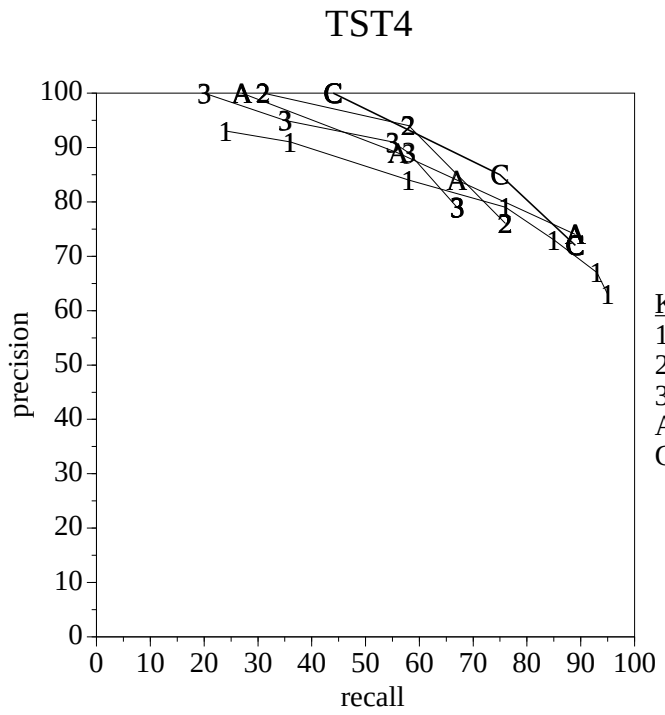
On the other hand, the augmented relevancy signatures algorithm did nearly as well, and the case-based algorithm did better. Furthermore, the augmented relevancy signatures algorithm and the case-based algorithm did much better than Rel-Sigs-2 and Rel-Sigs-3 on TST3. Our conclusion is that the augmented relevancy signatures algorithm and the case-based algorithm are the best approaches for classifying texts with high precision. Using multiple relevancy signatures may be



Algorithm	Recall	F(2)	F(1)	F(.5)	F(.3)	F(.2)	Precision
Rel-Sigs-1	91 79	91 79	90 83	77 90	67 94	67 94	30 100
Rel-Sigs-2	64 86	64 86	64 86	61 89	52 92	30 100	30 100
Rel-Sigs-3	48 94	48 94	48 94	48 94	48 94	48 94	23 100
Aug Rel Sigs	81 93	81 93	81 93	81 93	62 100	62 100	62 100
CBR	81 85	81 85	77 91	68 98	61 100	61 100	61 100

Figure 4.15: TST3 results for multiple relevancy signatures

effective as a stronger filter for relevant texts in some cases, but they also run the risk of decreasing recall levels.



Key
1 = 1 Relevancy Signature
2 = 2 Relevancy Signatures
3 = 3 Relevancy Signatures
A = Augmented Relevancy Signatures
C = Case-Based Algorithm

Algorithm	Recall	F(2)	F(1)	F(.5)	F(.3)	F(.2)	Precision
Rel-Sigs-1	95 63	93 67	85 73	76 79	58 84	36 91	24 93
Rel-Sigs-2	76 76	76 76	76 76	58 94	58 94	31 100	31 100
Rel-Sigs-3	67 79	67 79	67 79	58 89	55 91	35 95	20 100
Aug Rel Sigs	89 74	89 74	89 74	67 84	56 89	27 100	27 100
CBR	89 72	89 72	89 72	75 85	44 100	44 100	44 100

Figure 4.16: TST4 results for multiple relevancy signatures

4.8 Summary

In this chapter, we presented:

- Three text classification algorithms that are based on information extraction: the relevancy signatures algorithm, the augmented relevancy signatures algorithm, and a case-based text classification algorithm.
- Results in the terrorism and joint venture domains which show that IE-based text classification can achieve high precision with strong levels of recall. The IE-based text classification algorithms achieved 100% precision with 62% recall on one set of terrorism texts and 94% precision with 56% recall on the joint ventures corpus.
- Results in the terrorism domain which show that using increasing amounts of linguistic information can improve performance in some domains.
- Results in the microelectronics domain which suggest that keyword-based approaches may be more well-suited for some technical domains.
- A procedure for automatically deriving appropriate threshold values using a training corpus.
- Experiments with multiple relevancy signatures which suggest that using multiple indexing terms does not necessarily improve performance. In some cases, it can improve precision but in other cases it results in substantial recall loss. In all cases, using single indexing terms with richer representations (augmented relevancy signatures or cases) demonstrated better performance than multiple relevancy signatures.

CHAPTER 5

CHARACTERISTICS AND REQUIREMENTS FOR NEW DOMAINS

This chapter addresses issues concerning the general applicability of AutoSlog and the text classification algorithms. The first part of the chapter characterizes the types of domains and tasks that are appropriate for AutoSlog and the text classification algorithms. We also discuss the relative strengths and weaknesses of the three text classification algorithms. The second part of the chapter addresses practical concerns about portability. We outline the steps necessary to port these systems to new domains, including how to generate an appropriate training corpus and how to generate a semantic feature dictionary with miminal effort.

5.1 Characterizing Domains and Tasks

AutoSlog and the text classification algorithms performed well in the MUC-4 terrorism domain and the MUC-5 joint ventures domains but they did not perform as well in the MUC-5 microelectronics domain. In general, it is important to try to understand when techniques will work and when they will not. There are two dimensions that largely determine the effectiveness of information extraction and text classification techniques: the domain and the task. Up until now, we have not distinguished these factors. For example, we have been referring to the MUC-4 domain of terrorism, but we really should have separated the problem into two components: a domain description (terrorism), and a task description (the MUC-4 guidelines that define the specific aspects of terrorism that are relevant). In this section, we distinguish domains from tasks and define properties of each that determine which text processing techniques are appropriate.

5.1.1 Properties of Domains

We will refer to a *domain* as a general subject matter or topic area. For example, terrorism, joint ventures, and microelectronics are domains. Medicine is a domain. Herpetology is a domain. Different people undoubtedly have different definitions of some domains (e.g., what is terrorism?) but, in general, most people agree on a set of concepts that are central to a domain even though they may disagree on boundary cases.

We have identified two specific types of domains: technical and event-based. Technical domains are characterized by jargon and self-contained concepts; microelectronics and medicine are examples of technical domains. Table 5.1 shows several properties typically associated with technical domains. Technical domains are usually dominated by specialized noun phrases, e.g., "chemical vapor deposition" or "myocardial infarction". For the most part, specialized terms are

Table 5.1: Properties of technical domains

1. **specialized terms or jargon** specialized terms are usually noun phrases that represent complex concepts.
2. **self-contained concepts** specialized terms are typically unambiguous and do not depend on surrounding context.
3. **static vocabulary** relatively fixed set of relevant terms; idiosyncratic phrases are uncommon.

unambiguous and their meaning is not affected by surrounding context. The specialized nature of technical domains also implies that they have relatively static vocabularies. That is, a fixed set of terms are used to express most relevant concepts; idiosyncratic expressions and creative uses of language are rare.

In contrast, event-based domains revolve around actions and objects that play roles in the actions. Event-based domains are characterized by dynamic vocabularies and objects that acquire their semantics through role relationships. Terrorism and joint ventures are examples of event-based domains. Table 5.2 shows common properties of event-based domains.

Table 5.2: Properties of event-based domains

1. **event-based nature** concepts revolve around actions; verbs and verb nominalizations represent important domain concepts.
2. **contextual semantics** relevant objects acquire their semantics through role relationships.
3. **dynamic vocabulary** no fixed vocabulary; idiosyncratic expressions and context can express relevant concepts.

Event-based domains revolve around actions that are typically expressed with verbs or verb nominalizations. For example, terrorism is an event-based domain because it centers around events such as bombings and kidnappings. Bombings and kidnappings are often expressed with verbs (e.g., "bombed", "exploded", "kidnapped") and verb nominalizations (e.g., "bombings", "abductions"). The objects involved in relevant events are defined by their roles. That is, relevant objects often acquire their semantics through role relationships. For example, a victim is defined a person who plays a specific role in a terrorist event. It is impossible to identify a person as a victim solely by their name or referent (i.e., "Stanley Bingham" or "the man"). The context completely determines the role of the person; for example, "the man was kidnapped" implies that "the man" is a victim but "the man kidnapped the child" implies that "the man" is a perpetrator. Similarly, it is impossible

to identify a company as a joint venture partner merely by its name. The surrounding context determines whether or not the company is a partner in a joint venture.[1]

Event-based domains typically do not have fixed vocabularies. Relevant concepts are frequently expressed with idiosyncratic phrases, for example the sentence "the guerrillas put an end to the life of the mayor" describes a murder. Events can also be expressed solely by context rather than individual words or phrases. For example, the sentence "the building suffered damage as a result of a bomb planted by the FMLN" describes a bombing. Although the words "bomb" and "planted" are indicative of a bombing, the fact that the building was damaged is essential to recognize the event as an actual bombing instead of an attempted bombing. Similarly, the following sentence describes a joint venture: "GM will begin producing a new sedan which is the result of a collaboration with Toyota". This sentence describes a joint venture because it mentions two companies that are collaborating together to produce a new product. Although "collaboration with Toyota" is a good indicator of a joint venture, the fact that the two companies are producing a product is a crucial piece of information that distinguishes the event from other types of collaborations. Event-based domains may also have strong keywords and phrases but, in general, they have dynamic vocabularies that are open-ended in theory and in practice.

5.1.2 Properties of Tasks

The effectiveness of a text processing system depends crucially on the definition of the task as well as the domain. For example, there are many possible criteria for classifying texts in the joint ventures domain, a few of which are listed below. A text might be considered relevant if it contains:

- any reference to joint ventures in general.
- a reference to a specific joint venture between named partners.
- a reference to a specific joint venture between named partners, one of which is Japanese, to produce automobiles.
- a reference to a specific joint venture between two named partners, one Japanese company and one American company, within the last 6 months, to produce subcompact cars or minivans, for which the total capitalization exceeds 50 million dollars.

A task definition can be arbitrarily simple or complex. The properties of the task must be considered in order to determine which technique is most appropriate. Very generally, the goal of information extraction and text classification systems is to identify relevant information in text. The task definition implicitly or explicitly determines what types of information a system must be able to recognize. Table 5.3 lists several types of information that require different text processing capabilities.

Table 5.3 is not meant to be a comprehensive list of all possible information types, but is intended merely to distinguish some common information types. In the following sections, we explain which domains and information types are appropriate for AutoSlog and the text classification algorithms.

---

[1]This can be true of technical domains as well; for example, a material can be a component or an ingredient. But this distinction usually revolves around how an object is being *used*, which reflects some sort of action or process.

Table 5.3: Six common information types in task descriptions

Technical Information: Technical terms, scientific jargon, proper nouns. For example: microelectronics devices ("x-ray steppers", "molecular beam epitaxy"), medical terms ("myocardial infarction"), automobile models ("Nissan Sentra"', "NX-7").
General Event References: Abstract references to a type of event or events. For example, generic references (e.g., "The United States is introducing legislation to encourage joint ventures with Russia.") and summary event descriptions (e.g., "There have been many bombings and kidnappings in the last year").
Specific Event References: Direct references to a *specific* event or set of events. For example, "Toyota formed a joint venture with Nissan" or "the back-to-back murders of the mayor and his successor".
Role Objects: Objects that acquire their semantics through role relations. For example, perpetrators and victims are people who play a specific role in an event.
Taxonomic Relations: Relationships between objects that are based on a semantic hierarchy of the world. For example, the fact that cats are a type of animal and that Bogota is a city in Colombia.
Computed Relations: Relationships that must be computed based on the value of an object. For example, comparing two monetary figures or the time interval between two dates.

5.2 Domains and Tasks for AutoSlog

5.2.1 Domains for AutoSlog

AutoSlog's primary function is to create concept node definitions to extract *information related to events*. For example, AutoSlog can generate definitions to extract the names of perpetrators and victims but it cannot generate definitions to extract the names of people in general. Similarly, AutoSlog can generate definitions to extract the names of companies involved in a joint venture, but it cannot generate definitions to extract the names of companies in general. *AutoSlog was designed to extract information that corresponds to roles in an event.*

Consequently, AutoSlog is most effective in event-based domains. The results in Sections 3.5.1 and 3.5.2.5 support this claim by demonstrating that AutoSlog performed well in the terrorism and joint ventures domains, which are both event-based in nature. In contrast, Section 3.5.3.1 described some difficulties with AutoSlog in the microelectronics domain, a technical domain. Since technical information is usually self-contained in noun phrases, the surrounding context rarely contains words that are useful for identifying the technical information itself. It follows that it is not necessary to use concept nodes to extract technical information; searching for key words and expressions is simpler and more efficient. However, as we explain in the next section, AutoSlog can be useful for technical domains with certain types of task descriptions when combined with keyword recognition
.

Table 5.4: Appropriate domains and techniques for information extraction

Domain Type\Technique	Keywords	AutoSlog
Technical Domains	√	
Event-Based Domains		√

5.2.2 Tasks for AutoSlog

AutoSlog is not useful for extracting technical information in general. If an information extraction task dictates that all references to x-ray stepper systems should be extracted, then simple keyword analysis will suffice. Simply put, if an information extraction task requires only the ability to extract technical information then AutoSlog is not necessary.

However, AutoSlog *is* useful for extracting technical information in event contexts. For example, if an information extraction task dictates that x-ray stepper systems should be extracted only in manufacturing contexts, then the task requires more than just the ability to recognize technical information. The information extraction system must be smart enough to extract the desired information only in the context of certain events. The MUC-5 microelectronics task falls into this category because microelectronics information was supposed to be extracted only in the context of five event types: developing, manufacturing, distributing, purchasing, or using (see Section 2.3.2).

In Section 3.5.3.1, we described how AutoSlog's definitions were combined with keyword recognition to extract technical information in specific contexts. First, AutoSlog generated concept node definitions to represent patterns for relevant event types (manufacturing, developing, etc.). Second, keyword recognition was applied in a second pass to ensure that only the desired technical information was extracted by the concept nodes; irrelevant information extracted by the concept nodes was thrown away.

AutoSlog's capabilities are best understood by consulting the list of information types in Table 5.3. AutoSlog can recognize event references indirectly. Concept node definitions often represent linguistic expressions that are associated with specific event types. For example, "bombing of", "kidnapped by", "formed venture", etc. These concept nodes can be used to identify event references, for example using the relevancy signatures algorithm.

AutoSlog was designed to extract role objects so these are, in some sense, AutoSlog's specialty. However, AutoSlog is not capable of recognizing taxonomic and computed relationships. For example, a task description might specify that all birds should be extracted or that all monetary amounts greater than one million dollars should be extracted. AutoSlog does not have knowledge that would enable it to identify general types of objects or to compute functions. Of course, domain-specific routines could be inserted into AutoSlog.[2] But, in the absence of external domain knowledge, AutoSlog's strength rests in its ability to generate definitions that recognize event references and role objects.

5.3 Domains and Tasks for Text Classification

5.3.1 Domains for IE-Based Text Classification

All three of the text classification algorithms described in this dissertation are based on an underlying information extraction system, so the limitations of the information extraction

[2]For example, routines could be inserted in a concept node's enabling conditions or slot filling constraints.

Table 5.5: Appropriate tasks and techniques for information extraction

Task Type\Technique	Keywords	AutoSlog
Technical Information	√	
General Event References	√	√
Specific Event References		√
Role Objects		√
Taxonomic Relations		
Computed Relations		

system are limitations of the text classification algorithms. Therefore technical domains are not particularly well-suited for IE-based text classification. Classifying texts based solely on the presence of absence of technical information is probably more easily and effectively handled by keyword systems.

However, IE-based text classification is appropriate for event-based domains. The underlying information extraction system can recognize event references and role objects which can then be used to classify texts. The effectiveness of the three text classification algorithms described earlier depends largely on the information types involved in the classification task. In the next section, we explain the strengths and weaknesses of the three text classification algorithms with respect to different task descriptions.

Table 5.6: Appropriate domains and techniques for text classification

Domain Type\Technique	Keywords	Rel Sigs	Aug Rel Sigs	CBR
Technical Domains	√			
Event-Based Domains		√	√	√

5.3.2 Tasks for IE-Based Text Classification

5.3.2.1 Tasks For Relevancy Signatures

As we explained, technical information (scientific terms, proper names, etc.) can be handled by keyword systems, unless the task involves classifying texts on the basis of technical information in certain contexts. Within the realm of event-based domains, *general event references* are also well-suited for keyword systems because they can often be recognized by looking for the presence or absence of key words or phrases that refer to the event. General event references are defined as *any* reference to an event, abstract or specific, in any context. Relevancy signatures can be used to identify general event references, but it is not clear that they are have any substantial advantage over keywords for this task. The main advantage of the relevancy signatures algorithm is that its corpus-based approach can suggest good indexing terms that might not have been considered otherwise. However, if there are obvious key phrases and expressions that are useful for the domain then the additional overhead associated with generating a training corpus may not be worth relatively small performance improvements.

On the other hand, *specific event references* require more than just keyword analysis. In many cases, the presence of a role object is a key factor in distinguishing specific events from abstract

events. For example, expressions such as "the murder of the mayor" and "Nissan and Toyota formed a joint venture" refer to specific events but "murders and bombings are commonplace in Bogota" and "a company must invest millions of dollars to start a joint venture" are abstract references to events.

One expects keyword techniques to do well with the joint ventures domain because it has many strong key words and phrases, such as "joint venture", "tie-up", etc. Table 5.7 shows recall and precision scores obtained by retrieving all texts from the JV corpus that contain the word(s) in the left-hand column. Not surprisingly, 95.5% of the relevant texts contain the word "venture" and 93.3% of them contain the word "joint" and the word "venture".[3] This confirms the intuition that keywords are capable of achieving high recall in the joint ventures domain. In the precision column, 88.9% of the texts that contain both the word "joint" and "venture" are relevant. These results confirm that keywords can obtain good precision in this domain.

Table 5.7: Recall and precision scores for selected JV words

Words	**Recall**	**Precision**
joint, venture	93.3%	88.9%
tie-up	2.5%	84.2%
venture	95.5%	82.8%
jointly	11.0%	78.9%
joint-venture	6.4%	73.2%
consortium	3.6%	69.7%
joint, ventures	19.3%	66.7%
partnership	7.0%	64.3%
ventures	19.8%	58.8%

However, some of the results in the table are less impressive. Except for the word "tie-up", none of the other words obtain greater than 80% precision. Even the hyphenated word "joint-venture" gets only 73.2% precision. Furthermore, only 58.8% of the texts that contain the word "ventures" are relevant. This is the same phenomena that we saw in the terrorism domain (see Section 4.3.2). The singular form "venture" is often used to describe a specific event (e.g., "a venture between X and Y") but the plural form is frequently used in generic or summary descriptions (e.g., "China and the United States have been involved in many joint ventures."). Some words that seem useful intuitively achieve good precision but do not exceed 90% precision (e.g., "venture", "tie-up") and others get rather low precision (e.g., "ventures", "consortium", "partnership") . A user would find it difficult if not impossible to anticipate which words are useful and which ones are not.

Relevancy signatures capture only minimal context surrounding individual words, but a small amount of linguistic information can make a big difference. Table 5.8 shows the relevancy percentages associated with some signatures for the joint ventures domain.[4,5] Note that several

[3]Not necessarily in adjacent positions but somewhere in the text.

[4]These statistics were generated from the training texts in the first fold of the cross-validation experiments described in Section 4.6.3.3.

[5]Instead of the actual signatures, which are quite long, Table 5.8 shows the AutoSlog pattern recognized by each signature.

signatures associated with the word "venture" achieve higher precision[6] than the "joint venture" keyword query. In particular, the signatures representing "venture between" and "venture with" achieve 100% and 95% precision respectively. The strange-looking pattern, "<entity> <verb> venture", represents all occurrences of the word "venture" preceded by any verb.[7] This pattern achieved better precision (93%) than keywords alone. The presence of certain prepositions (e.g., "between" and "with") or a preceding verb produces more accurate classifications than the word "venture" by itself, presumably because they are indicative of *specific* incidents. Similarly, the signature for the expression "tie-up with" achieves 100% precision whereas the word "tie-up" alone achieved only 84.2% precision.

Table 5.8: Relevancy percentages for selected signatures

Signature Pattern	**Precision**
venture between <entity>	100.0%
venture with <entity>	95.9%
venture of <entity>	95.4%
<entity> <verb> venture	93.0%
venture by <entity>	90.9%
tie-up with <entity>	100.0%
<entity> form company	100.0%
<entity> set up company	100.0%
<entity> join forces	100.0%
<entity> agreed to form	100.0%
project between <entity>	100.0%
project with <entity>	75.0%
<entity> construct	100.0%
<facility> was constructed	63.6%
set up with <entity>	94.7%
set up by <entity>	66.7%
deal with<entity>	56.8%
cooperation with <entity>	50.0%

Another advantage of relevancy signatures is that they can recognize expressions that are highly correlated with relevance even though none of the constituent words are highly correlated with the domain. For example, the phrases "form company", "set up company", "join forces", and "agreed to form" all achieved 100% precision. It may be difficult for users to think of many useful expressions when constructing a query, and to know which ones are truly reliable. Table 5.8 shows some striking differences between similar signatures. For example, "project between" is highly

[6]Strictly speaking, these are the estimated conditional probabilities of the signatures. However, for the sake of comparison with the keywords, we will refer to them as precision results that would have been achieved by using each signature to classify the training texts.

[7]The concept node for this pattern was created by the generalization modules described in Section 3.5.2.4.

relevant but "project with" is not; the active form of the verb "construct" achieves high precision but the passive form "was constructed" does not[8]; "set up with" is highly correlated with relevance but "set up by" is not. Finally, two expressions that seem like they should be related to joint venture activities, "deal with" and "cooperation with", are *not* highly correlated with joint venture texts.

The power of the relevancy signatures algorithm comes from its ability to recognize expressions that contain slightly more context than keywords, and from the relevancy correlations generated automatically from a training corpus. Local linguistic context is often enough to distinguish specific events from abstract event references. Relevancy signatures can infer the presence or absence of role objects by virtue of prepositions (e.g., "tie-up with" implies that another partner is mentioned) and verb forms ("was kidnapped" implies that a victim is mentioned).

5.3.2.2 Tasks For Augmented Relevancy Signatures

The relevancy signatures algorithm is the simplest algorithm to implement because it does not depend on semantic features. For the same reason, however, it is the least capable of making contextual distinctions. However, some task descriptions specify that only certain semantic classes of role objects are legitimate for a given domain. For example, in the MUC-4 task, an event was considered to be terrorist only if it involved civilian victims or targets. Attacks by terrorists on military personnel or military targets were *not* relevant. Similarly, attacks by terrorists against other terrorists were not relevant.

The augmented relevancy signatures algorithm can recognize the semantic classes of role objects. For example, Section 4.4.3 showed that the augmented relevancy signatures algorithm performed better than the relevancy signatures algorithm in the MUC-4 domain of terrorism because of its ability to distinguish between military and non-military targets. Table 5.9 shows the 20 slot triples that were chosen as "reliable" slot triples based on 1500 texts[9] using a relevancy threshold of 95% and a frequency threshold of 10.[10] The first component of each triple represents the event type, the second component represents the role, and the third component represents the semantic type of the role object. For example, the triple (arson, perpetrator, TERRORIST) represents terrorist perpetrators of arson events.

Table 5.9 shows that many types of civilian targets were highly correlated with relevant texts. For example, when COMMERCIAL and FINANCIAL buildings (e.g., stores and banks) are attacked it was almost always a terrorist incident. The triple (bombing, instr, VEHICLE-BOMB) shows that every bombing that used a VEHICLE-BOMB (i.e., car bomb or truck bomb) came from a relevant text. When a COMMERCIAL, COMMUNICATIONS, FINANCIAL, or GOVT-OFFICE-OR-RESIDENCE buildings was destroyed, it was always in a relevant text. The loc-val event type represents the concept of placing or planting a device; for example, loc-val concept nodes are activated by phrases such as "was planted", "was placed", "X planted Y", etc. Table 5.9 shows that when bombs, dynamite, or explosives are planted, it is almost always in a terrorist text. The table also shows that all texts in which a judge is threatened, (threat, victim, LEGAL-OR-JUDICIAL), are relevant texts.[11]

---

[8]Presumably this is because the active form recognizes the future tense, i.e. "will construct"

[9]The MUC-4 development corpus plus TST3 and TST4.

[10]That is, at least 95% of the occurrences of the triple came from relevant texts and there were at least 10 occurrences in the corpus.

[11]Table 5.9 also has some circumstantial entries. The triple (murder, victim, LOCATION) is the result of an idiosyncracy in the MUC-4 dictionary. The MUC-4 corpus contains many stories about

To summarize, augmented relevancy signatures are appropriate for tasks that require the ability to distinguish between different classes of role objects.

Table 5.9: Relevancy percentages for reliable slot triples

Slot Triple	Precision
(arson, perpetrator, TERRORIST)	100.00%
(attack, target, COMMERCIAL)	95.65%
(attack, target, FINANCIAL)	100.00%
(attempted-bombing, instr, BOMB)	100.00%
(bombing, instr, EXPLOSIVE)	100.00%
(bombing, instr, VEHICLE-BOMB)	100.00%
(bombing, target, COMMERCIAL)	100.00%
(bombing, target, DIPLOMAT-OFFICE-OR-RESIDENCE)	100.00%
(bombing, target, ENERGY)	95.00%
(destruction, target, COMMERCIAL)	100.00%
(destruction, target, COMMUNICATIONS)	100.00%
(destruction, target, FINANCIAL)	100.00%
(destruction, target, GOVT-OFFICE-OR-RESIDENCE)	100.00%
(injury, perpetrator, TERRORIST)	100.00%
(loc-val, instr, BOMB)	95.92%
(loc-val, instr, DYNAMITE)	100.00%
(loc-val, instr, EXPLOSIVE)	100.00%
(loc-val, perpetrator, TERRORIST)	95.46%
(murder, victim, LOCATION)	95.65%
(threat, victim, LEGAL-OR-JUDICIAL)	100.00%

5.3.2.3 Tasks For Case-Based Text Classification

The strength of the augmented relevancy signatures algorithm is its ability to recognize the semantic classes of role objects. However, it can recognize only one role object at a time. The case-based algorithm can consider multiple role objects simultaneously by virtue of its *case outline*, which is one component of the relevancy indices used by the algorithm. As we described in Section 4.5.3, a case outline represents the set of roles that were identified in a sentence. For example, the case outline for the sentence "FMLN guerrillas killed the man using a machinegun" is

---

the kidnapping and murder of a woman named Gilda Flores. However, Flores is also the name of a city in El Salvador and Guatemala so it is tagged as both a location and a proper-name in the dictionary. Locations have their own semantic feature which is distinct from other types of proper names. Whenever the word "Flores" was extracted as a murder victim it was almost always a reference to Gilda Flores and was therefore describing a relevant incident. Although this triple does not make much sense in general (locations aren't usually murder victims!), it is not likely to cause much trouble because murder concept nodes will not normally extract locations anyway.

(victims perpetrators instruments) because the sentence contains victims ("the man"), perpetrators ("FMLN guerrillas"), and instruments ("a machinegun").

Therefore, the case-based algorithm can classify texts on the basis of multiple role objects. For example, many texts in the MUC-4 corpus contained summary descriptions such as "Hundreds of Salvadorans were murdered last year.". These summary descriptions were misleading because they referred to civilian victims of violent crimes. However, *specific* incidents almost always mentioned a perpetrator, e.g. "Hundreds of Salvadorans were murdered by the FMLN today.". The case-based algorithm was able to distinguish these events by looking for the presence of a civilian victim and a known perpetrator.[12]

The case-based algorithm also has the property that it can recognize event descriptions solely by context. If a domain does not have strong key phrases then it will not be amenable to the other algorithms. For example, one might want to identify texts that mention a violent crime involving stolen property. A sentence such as "A man threatened the tourist and took his wallet" would not be recognized by the other algorithms because it does not contain a strong key phrase. But the case-based algorithm can recognize it by looking for the presence of multiple objects (perpetrator and stolen object). If a domain *does* have strong key phrases (e.g., the joint ventures domain), however, then the case-based algorithm can be at a disadvantage because it does not require the presence of a key phrase.

To summarize, the case-based algorithm is appropriate for tasks that require the presence of multiple role objects. The case-based algorithm represents the semantic class of only one object, but can recognize the presence or absence of others. Domains without strong key phrases and task descriptions with multiple parameters are particularly well-suited for the case-based approach.

Table 5.10: Appropriate tasks and techniques for text classification

Task Type\Technique	Keywords	Rel Sigs	Aug Rel Sigs	CBR
Technical Information	√			
General Event References	√	√		
Specific Event References		√	√	√
Role Objects		√	√	√
with semantics			√	√
multiple objects				√
Taxonomic Relations				
Computed Relations				

5.4 Moving to New Domains

We have claimed throughout this dissertation that AutoSlog and the text classification algorithms are portable across domains. However, they rely on two important resources: the CIRCUS sentence analyzer and a training corpus. In this section, we briefly describe how to acquire the lexicons that CIRCUS needs and how to generate appropriate training corpora.

[12]There are other ways to make this distinction, such as looking for date ranges (e.g., today vs. last year). Other approaches might be more reliable but they usually require hand-coded domain knowledge.

5.4.1 Resources Required for CIRCUS

CIRCUS relies on a lexical dictionary of part-of-speech tags for common and domain-specific words. Since CIRCUS is a selective sentence analyzer, it does not necessarily need to know the part-of-speech of every word in a text. However, it is important for CIRCUS to have good coverage on the words that trigger concept nodes and words that should be extracted by concept nodes. Machine-readable dictionaries are readily available that contain part-of-speech tags for thousands of words. Alternatively, several statistically-based part-of-speech taggers have been developed that assign part-of-speech tags based on statistics generated from a corpus (e.g., OTB [Lehnert *et al.*, 1993a], PARTS [Church, 1989], and POST [Weischedel *et al.*, 1993]). The part-of-speech dictionary is an integral part of CIRCUS so it is required for AutoSlog and all of the text classification algorithms.

CIRCUS also has the ability to use a semantic feature dictionary, if one is available. Neither AutoSlog nor the relevancy signatures algorithm use semantic features at all.[13]. But the augmented relevancy signatures algorithm and the case-based algorithm do rely on a semantic feature dictionary. Some techniques have been developed to acquire semantic features dynamically during text processing (e.g., see [Cardie, 1993]). Alternatively, semantic features for many domain-specific words can be acquired with minimal effort by exploiting the annotated training corpus used by AutoSlog. In Section 4.6.3.2 we reported that it took several hours to build a semantic feature dictionary for the joint ventures domain using this approach.

5.4.2 Preparing a Training Corpus for Text Classification

The classification algorithms rely heavily on a domain-specific training corpus of texts. Since the statistics identify associations between extracted information and relevance, the training corpus must contain a good mix of both relevant and irrelevant texts. If the corpus contains mostly relevant texts, then virtually every phrase or context will be highly correlated with relevant texts. If the corpus contains mostly irrelevant texts then few, if any, phrases or contexts will be highly correlated with relevant texts. The ideal training corpus should be relatively balanced between relevant and irrelevant texts.

Furthermore, the irrelevant texts should be similar in nature to the relevant texts. That is, they should contain the same "types" of information. For example, the irrelevant texts in the MUC-4 corpus typically describe military incidents, civilian crimes, or terrorist activity in general. Since these stories frequently refer to violent crimes, perpetrators, victims, targets, and weapons, they activate many of the same concept nodes as the relevant texts. This is important because the statistical techniques need both positive and negative examples to identify reliable phrases and contexts. If the irrelevant texts do not activate any of the same concept nodes as the relevant texts, then every piece of extracted information will be highly correlated with relevance.

We should emphasize that these algorithms may not be appropriate for general corpora. As we explained in Section 2.2, the MUC-4 corpus was pre-filtered for the domain of terrorism. The texts were selected by keyword search from a database of newswire articles because they contain words associated with terrorism. The MUC-4 corpus satisfied both of the criteria described above: (1) about 50% of the texts were relevant and (2) the irrelevant texts were similar in nature to the relevant texts. This two-stage process shows how traditional IR systems can be pipelined with natural language processing systems to exploit the benefits of both approaches. Traditional IR techniques such as keyword filtering can be used to retrieve texts of a certain nature from a general

---

[13]In theory, relevancy signatures do not use semantic features, but some of the hand-crafted concept nodes used in these experiments did rely on semantic features in their enabling conditions. However, the concept node definitions automatically produced by AutoSlog do not use semantic features in their enabling conditions.

corpus. Then natural language processing techniques can be applied to this pre-filtered corpus to generate more accurate classifications. In Section 4.6.3.1 we described how we generated a training corpus for the joint ventures domain by combining relevant texts with irrelevant texts that were retrieved by traditional IR systems.

5.4.3 Annotating a Corpus for AutoSlog

As input, AutoSlog needs a training corpus of relevant texts in which the domain-specific information that should be extracted has been tagged with semantic labels. For example, in the domain of terrorism, the names of all perpetrators, victims, targets, and weapons that were involved in a relevant terrorist event should be tagged with their semantic type (e.g., VICTIM) and the event type (e.g., KIDNAPPING). NLP systems often rely on other types of tagged corpora, such as part-of-speech tagging or phrase structure bracketing (e.g., the Brown Corpus [Francis and Kucera, 1982] and the Penn Treebank [Marcus *et al.*, 1993]). However, corpus tagging for automated dictionary construction is less demanding than other forms of tagging because it is smaller in scope. For syntactic tagging, every word or phrase must be tagged but, for AutoSlog, only the targeted information needs to be tagged. Sentences, paragraphs, and even texts that are irrelevant to the domain can be effectively ignored.

All of the AutoSlog experiments described so far used manually encoded key templates from MUC4 and MUC-5 as input. However, we claimed in Section 3.3 that AutoSlog requires only an annotated corpus. The key templates are very time-consuming to create and contain a lot of information that AutoSlog does not need and, in fact, did not use. We took advantage of the MUC-4 and MUC-5 key templates as input to AutoSlog because we had them available to us. But in a new application a user would need to generate an annotated training corpus from scratch.

As input, AutoSlog needs to know which noun phrases in a text are relevant to the domain and should be extracted. To annotate a corpus for AutoSlog, a user needs to do three things:

1. identify relevant texts
2. identify relevant noun phrases that should be extracted from the text
3. assign semantic tags to the relevant noun phrases

Identifying relevant texts is required for the text classification training corpus as well as for AutoSlog's training corpus. In practice, we envision that a traditional information retrieval system would be used to identify several hundred texts from a general corpus that are good candidates for the domain; presumably, most of the retrieved texts will be either relevant or "near-miss" texts. A user would then manually sift through the retrieved texts and separate the relevant from the irrelevant texts.

After the relevant texts have been identified, relevant noun phrases in the text must be marked so that AutoSlog knows which pieces of information it should focus its efforts on. In the NLP lab at the University of Massachusetts, we have a user-friendly text marking interface that allows a user to skim a text, highlight one or more words with the mouse, and assign semantic labels to the selected item by clicking a few buttons. In this manner, a user can quickly scan a corpus of texts and easily assign semantic annotations to relevant noun phrases.

To illustrate this process, Figure 5.1 shows an example sentence from the joint venture corpus and the appropriate semantic annotations. The sentence contains 5 pieces of information that should be extracted: 4 entities ("Japan Airlines Co.", "JAL", "Toyo Real Estate Co.", and "Sanwa Bank") and one facility ("hotel"). In the joint ventures domain, each noun phrase could be marked with up to 3 semantic labels. The first label specifies the general type of information (e.g., entity name, entity alias, or facility), the second label specifies the subtype (e.g., entities can be companies, governments, or people), and the third label specifies the role of the object (applies to entities only; e.g., an entity can be a parent or child company).

Figure 5.1: Example text annotations for AutoSlog

Each noun phrase that is marked will be given to AutoSlog as input. The interface keeps track of the location of the noun phrase in the sentence so AutoSlog does not need to search for the correct occurrence of the noun phrase.[14] AutoSlog then proposes a concept node definition that is capable of extracting the desired noun phrase. In a preliminary experiment with a novice user, the user annotated 160 texts using the text marking interface in about 8 hours.

5.4.4 How Much Training Data Does AutoSlog Need?

In the terrorism domain, AutoSlog generated an effective concept node dictionary using a training corpus of 772 relevant texts and associated answer keys. In the joint ventures domain, AutoSlog produced a good dictionary using a training corpus of 924 relevant texts and associated answer keys. But perhaps a smaller training corpus would been sufficient. Or perhaps a larger training corpus would have generated a much better dictionary. It would be useful to have some idea of how large a training corpus AutoSlog needs to produce a dictionary with good coverage of the domain.

To answer this question, we should identify which concept nodes are the most important ones for a domain. One possibility is to count the number of concept nodes that are kept by the human-in-the-loop. Presumably, the number of new concept nodes kept by the user will taper off

[14]For example, Japan Airlines Co. might be mentioned several times in many types of activities. When using the key templates as input, AutoSlog relied on the assumption that the first occurrence of the noun phrase was the relevant one. This assumption is not necessary with the text marking interface.

as training proceeds. However, not all of the concept nodes kept by the user are equally important. As we noted in Section 3.6.1, in the terrorism domain we found that 18% of the concept nodes produced 80% of the output. In other words, a small number of concept nodes do most of the work. Any dictionary that contains these important concept nodes will probably do reasonably well.

Ideally, we should calculate the percentage of correct information extracted by each concept node. But as we explained in Section 3.5.2.5, this can only be determined by manual inspection. Instead, we calculated how often each concept node "fired" in the corpus; that is, how often it extracted anything at all. If a concept node fired frequently, then it probably extracted a lot of relevant information although perhaps a lot of irrelevant information as well. If a concept node fired infrequently, then it didn't extract much information of any kind. The firing frequencies are not a perfect measure of importance, but they provide a general sense of how much influence each concept node has. For the sake of accuracy, we should note that we computed the firing frequencies of each concept node *pattern* instead of the individual concept nodes.[15]

Next, we partitioned the patterns into 19 groups depending upon which segment of the training corpus they were derived from. More specifically, we divided the training corpus into 19 groups of 50 texts each[16], put the patterns generated by the first 50 texts into partition #1, the patterns generated by the second set of 50 texts into partition #2, etc. The order of the partitions reflects how soon the pattern was generated by AutoSlog during training. For example, the patterns in partition #1 were created by the first 50 texts, the patterns in partition #2 were created by the first 100 texts, etc. Finally, we computed the percentage of the total firing frequencies accounted for by the patterns derived from the first 50 texts, the first 100 texts, the first 150 texts, etc.

Figure 5.2 shows the graph created by this analysis. This graph shows that the patterns acquired from the first 50 texts accounted for 42.5% of the total firing frequencies and the patterns acquired from the first 100 texts accounted for 53.8% of the firing frequencies. This result implies that after only 100 texts AutoSlog produced a dictionary with over half the functionality of the dictionary produced with 924 texts. After 300 texts, the patterns accounted for about 80% of the firing frequencies, and the patterns derived from the first 500 texts accounted for about 90% of all firings.

Ultimately, determining how many texts AutoSlog needs to generate a dictionary with good coverage depends on the nature of the domain, the corpus, and the coverage required. A domain with a general focus (e.g., medicine) will likely require more patterns than a domain with a very specific focus (e.g., leukemia). And the number of patterns that are required depends on how many different types of information need to be extracted. However, Figure 5.2 suggests that AutoSlog's training phase reflects a curve of diminishing returns. The most common patterns are acquired early in the training phase and, in general, patterns acquired later have less impact on the overall performance of the dictionary. When applying AutoSlog to a new domain, a good strategy for deciding when to stop training is to keep track of a firing frequency curve. As long as the slope of

---

[15]The reason is because the same pattern can be used to extract different types of information. For example, the pattern "venture with <x>" can extract both company names and people names. Different concept nodes representing the same pattern will always have identical firing frequencies (unless the slot constraints prevent certain items from being extracted). However, one of the concept nodes may represent a pattern that is much more common than the others. For example, the vast majority of joint venture partners are companies, not individual people. In this case, the firing frequencies would make it appear that the uncommon pattern appeared much more frequently than it actually did.

[16]There were 1000 texts in the MUC-5 corpus but AutoSlog used only the 924 relevant texts for training.

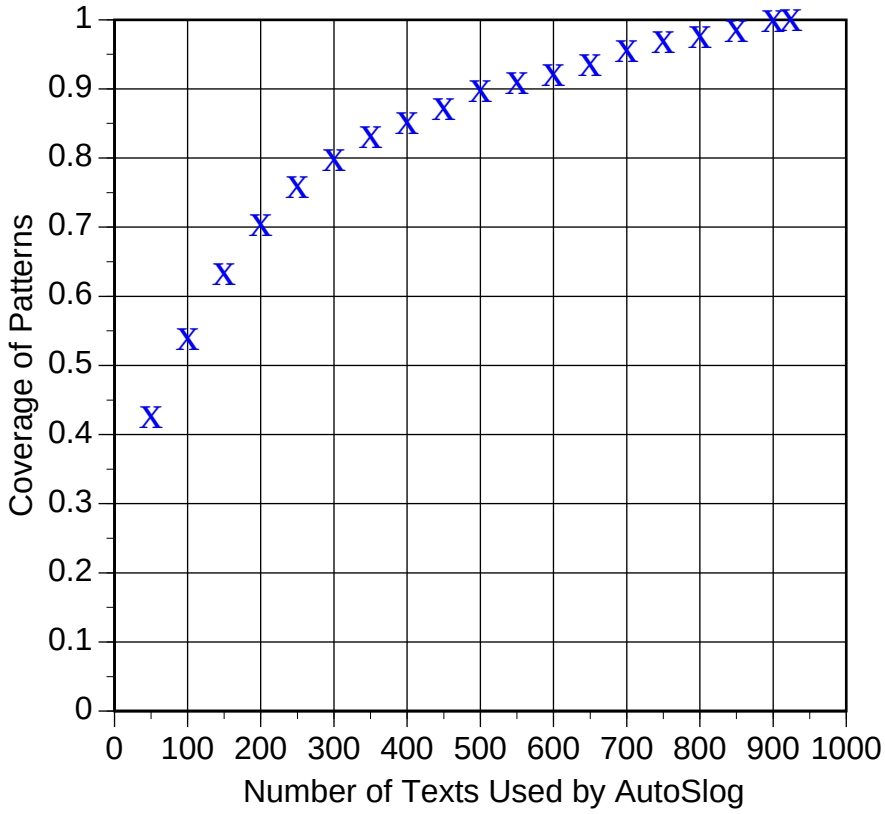


Figure 5.2: The relationship between dictionary coverage and training corpus size.

the curve is steep, AutoSlog is still producing patterns that contribute significantly to the overall output. However, when the curve begins to level off AutoSlog is producing patterns that do not contribute as much and the effect of additional training is likely to be small.

5.4.5 Run-Time Measurements

For all three algorithms, the time required for training and classification is dominated by the sentence analyzer, CIRCUS. The average text length in the MUC-4 corpus is 300 words; CIRCUS took approximately 13 seconds to process each text, on average. Once the training corpus of 1500 texts was processed by CIRCUS, it took roughly 5.1 minutes to generate the signature data and 1.3 minutes to generate the slot triple data. The case base took 12.7 minutes to build. Classifying new texts takes about 13 seconds per text for sentence analysis plus the time required by the algorithms for classification, which is negligible (2-4 seconds per 100 texts). All of these experiments were performed on a Decstation 5000/240 running Ultrix 4.2 with 64 MB of RAM.

5.5 Summary

In this chapter, we identified properties of domains and tasks that determine the effectiveness of AutoSlog and IE-based text classification. We also discussed issues and resources associated with moving the systems to new domains. We concluded that:

- In general, AutoSlog and IE-based text classification are well-suited for event-based domains but not technical domains.
- AutoSlog is appropriate for information extraction tasks involving event references and role objects. Keywords are most appropriate for tasks involving technical information and jargon.
- IE-based text classification is appropriate for tasks involving event references and role objects. Relevancy signatures have some advantages over keywords for recognizing event references and role objects. Augmented relevancy signatures have the ability to recognize semantic classes of role objects. The case-based algorithm has the ability to recognize multiple role objects.
- Porting AutoSlog to new domains requires a part-of-speech tagger or dictionary and an annotated training corpus of marked texts.
- Porting the text classification algorithms requires a training corpus of roughly 50% relevant and 50% irrelevant texts, where the irrelevant texts are "near-miss" texts. The augmented relevancy signatures algorithm and case-based algorithm also require a semantic feature tagger or dictionary.

CHAPTER 6

RELATED WORK

In this chapter, we describe how AutoSlog and the text classification algorithms relate to previous work. AutoSlog is most directly related to work in automated knowledge acquisition for natural language processing. AutoSlog can also be viewed as a machine learning program, so we will briefly discuss its relationship to machine learning techniques.

The problem of text classification has been addressed both in the AI community and in the information retrieval community. We also address the relationship of relevance feedback to text classification and discuss previous work on integrating natural language processing with information retrieval. Finally, we briefly discuss the case-based text classification algorithm in the context of the larger case-based reasoning community.

6.1 Automated Knowledge Acquisition for NLP

Automated knowledge acquisition is an important area of research which spans many subfields of artificial intelligence. In particular, several systems have been developed to acquire knowledge automatically for natural language processing systems. We describe some of these systems and compare them to AutoSlog.

FOUL-UP [Granger, 1977] was one of the earliest AI systems that automatically learned the meaning of unknown words. FOUL-UP was developed in the context of other knowledge-based natural language processing systems created at Yale in the late 1970s based on conceptual dependency (CD) theory [Schank, 1975]. When an unknown word was encountered, FOUL-UP inferred syntactic and semantic attributes for the word based on the expectations of a sentence analyzer, ELI [Riesbeck, 1978] and a script applier, SAM [Cullingford, 1978]. Nouns were assumed to fill an empty slot in the current CD frame (event) in the active script. Verbs were more complicated because they represent CD frames themselves. FOUL-UP assigned a generic CD frame to the unknown verb and then used a set of heuristics to map the CD frame into the script.

The POLITICS [Carbonell, 1979b] system also contained a mechanism for learning definitions of unknown words. First, POLITICS acquired syntactic and semantic information for an unknown word through "constraint projection". Constraint projection integrated syntactic expectations, semantic expectations from existing CD frames, and any other information known about the unknown word from elsewhere in the sentence. Second, POLITICS made inferences about the goals of the actors and recipients in the given situation and how the current action might help them achieve their goals. If these inferences produced additional information about the unknown word (e.g., a country is probably communist) then the new information was added to the word's definition.

Both FOUL-UP and POLITICS learned information about unknown words by examining contextual expectations derived from other words in the sentence. These systems relied heavily on knowledge structures (scripts and goal trees, respectively) that were manually encoded in the system for the domain. Expectation-based approaches to word learning are potentially powerful

techniques, but they are not yet practical because the domain knowledge needs to be manually encoded into the system.

RINA [Jacobs and Zernik, 1988] is a language acquisition system that uses multiple examples and a variety of knowledge sources to create dictionary entries for unknown words. For each unknown word, a text processing system called TRUMP creates a "hypothesis chart" that contains hypotheses about syntactic and semantic attributes of the unknown word. TRUMP uses morphology, syntax, conceptual semantics, lexical semantics, and discourse to infer properties about the word. A new dictionary entry is created by generalizing the hypotheses (variabilization, removing specifiers, etc.). Subsequent examples of the word are used to confirm or refute the hypotheses and add new information. Although the definitions produced by RINA are semantically rich, RINA relies heavily on the definitions of other words in the text, including lexical and conceptual information. In [Jacobs and Zernik, 1988], the authors admit that RINA worked on only a few examples. RINA is also sensitive to the order of examples and the specific linguistic clues in the text strongly influence the resulting definitions.

One of the systems most closely related to AutoSlog is PALKA [Kim and Moldovan, 1993]. PALKA automatically acquires phrasal patterns to extract information from text based on the MUC-4 terrorism domain. The output produced by PALKA looks a lot like the output produced by AutoSlog, but the mechanisms used to acquire the patterns are very different. As input, PALKA uses training texts, a set of generic frame definitions (e.g., bombing and kidnapping frames), and a set of keywords to trigger each frame. For each frame, PALKA identifies the sentences that contain keywords for the frame, parses these sentences, and extracts the clauses that contain the keywords. If a user verifies that the sentence is indeed relevant, then PALKA tries to map the segments of the clause to the slots in the frame using the semantic features associated with each word. If more than one mapping is possible then the MUC-4 templates are used to identify the correct mapping; if templates are not available then a user is consulted. Finally, a generalized pattern is created by replacing nouns with their semantic features and verbs with their root forms. If PALKA generates the same pattern more than once then their constraints are generalized and they are merged.

PALKA should be distinguished from AutoSlog along several dimensions. First, PALKA is given a set of generic frames and keywords up front. For example, it is told that a specific set of keywords should trigger a bombing frame. In contrast, AutoSlog discovers the trigger words for case frames on its own. Second, PALKA relies on the semantic features associated with words; AutoSlog does not need semantic features to generate concept nodes.

In summary, AutoSlog is different from other lexical acquisition systems because most previous systems depend on a "partial lexicon" as a starting point (e.g., [Carbonell, 1979b, Granger, 1977, Jacobs and Zernik, 1988]). These systems construct definitions for new words based on the definitions of other words in the sentence or surrounding context. AutoSlog builds new dictionary definitions completely from scratch and depends only on a part-of-speech lexicon, which can be readily obtained from machine-readable dictionaries (see Section 5.4.1).

One exception is recent work on automatically deriving knowledge from on-line dictionaries, described in [Dolan *et al.*, 1993, Montemagni and Vanderwende, 1992]. The work applies syntactic and lexical patterns to the entries in an on-line dictionary to derive semantic relationships between words. For example, verb definitions usually begin with an infinitive form that is a superclass of the verb, such as the following definition: **inquire**: "to ask for information". Similarly, noun definitions usually begin with another noun that is a superclass of the first, such as: **aster**: "a garden flower with a bright yellow center". The regularity in dictionary entries allows the system to extract semantic relationships from the dictionary and automatically build a semantic network (e.g., **aster isa flower**). More complicated patterns can derive structured information such as part-of relations and "typical" subjects and objects.

This work is similar to AutoSlog because syntactic rules are applied to natural language text to extract semantic relationships. Their results are encouraging because they lend independent support to the idea that semantic information can be acquired automatically without a lot of

external knowledge. Like AutoSlog, their system can also be viewed as a one-shot learning system, which we discuss in the next section.

6.2 Machine Learning

AutoSlog automatically generates concept node definitions from a training corpus, so in some sense it is a machine learning program. However, the method of learning used by AutoSlog does not fall neatly into any of the existing categories of machine learning research.

Since AutoSlog creates dictionary entries from scratch, it effectively does one-shot learning. The closest points of comparison in the machine learning community are explanation-based learning (EBL) systems [DeJong and Mooney, 1986, Mitchell *et al.*, 1986]. Explanation-based learning systems are known for their ability to produce complete concept representations from a single training instance. This is in contrast to inductive learning techniques that incrementally build a concept representation in response to multiple training instances (e.g., [Fisher, 1987, Quinlan, 1986, Utgoff, 1988]). Most inductive learning methods (e.g., decision tree algorithms [Quinlan, 1986, Utgoff, 1988]) take multiple training instances as input, both positive and negative instances of a concept, and use the instances collectively to build a representation of the concept (e.g., a decision tree). Having a good mix of both positive and negative training instances is necessary to produce a target representation that can recognize new instances of the concept (generalization) and discriminate negative instances of the concept (specialization).

Explanation-based learning systems take a single training instance as input (a positive example of the target concept) and attempt to explain *why* the example is a member of the concept. In effect, they generate a proof (explanation) showing that the example is indeed a member of the concept based upon the rules in the domain theory. The proof is generalized using simplified goal regression techniques. The goal concept is pushed back up through the proof structure; during this process, variables are unified and disjunctive subgoals are dropped. The result is a generalized explanation that represents the proof path for the training instance but can apply to new instances as well.

The primary difficulty with EBL systems is that they require an explicit domain theory. For most domains, an explicit domain theory is not readily available or practical to obtain. In addition, EBL is effectively a *deductive* learning technique because EBL systems never really learn anything that they didn't already know. They merely generate operational explanations that can be applied more efficiently to new instances (earlier work on macro operators [Fikes *et al.*, 1972] is based on the same idea).

AutoSlog does not rely on any explicit domain theory. Instead, it uses a set of general, domain-independent heuristics that are based on syntactic patterns in natural language. The input to AutoSlog is a text and noun phrase that needs to be extracted from the text (tagged with a semantic label, e.g. "bombing victim" or "jv company"). The output is a case frame representing a specific linguistic expression that can extract the noun phrase in the given text and will be able to extract similar noun phrases from future texts (e.g., bombing victims and joint venture companies). A key aspect of AutoSlog is that the input (noun phrases) is of a completely different form than the output (case frame).

One of the major distinctions between AutoSlog and other learning systems is that AutoSlog creates each case frame based solely on a single training example and no background knowledge. AutoSlog's general syntactic heuristics produce a separate case frame in response to each training example. Therefore each step of the concept node acquisition phase is completely independent from the others.[1] One of the advantages of this approach is that the ordering of the training

[1]With the exception that AutoSlog keeps track of the definitions it has already produced so that it doesn't generate duplicates.

examples does not matter. AutoSlog is an inductive learning system, however, because the case frames produced by AutoSlog produce a dictionary that can recognize expressions and extract information that it previously could not.

In summary, AutoSlog is a one-shot learning system that generates case frames for information extraction automatically using a training corpus of texts and targeted noun phrases. AutoSlog is similar to EBL techniques because they both are capable of learning from a single training instance. However, AutoSlog does not require an explicit domain theory. Furthermore, AutoSlog is an inductive learning system because it acquires case frames that recognize expressions that the system could not recognize before.

6.3 Text Classification

6.3.1 Traditional IR approaches

Text categorization has not traditionally been the focus of work in the information retrieval community, but some researchers have worked on automatic indexing for text categorization. One of the earliest text categorization systems was developed by Maron [Maron, 1961]. Using a training set of 260 technical abstracts and 32 categories, the author identified 90 words that "peaked" in one of the categories (i.e., the distribution of the word was not flat across all of the categories). To classify a new text, Maron identified which of the indexing terms appeared in the document and computed the conditional probability of the category given those words. The categories for a text were then ranked by the probability values and the top-ranking category was chosen. Maron evaluated the algorithm on a test set of 145 abstracts and reported that 20 contained none of the indexing terms, 40 contained only one term, and 51.8% of the remaining 85 texts were correctly classified.

Borko and Bernick followed up on Maron's work by using the same corpus and indexing terms to derive their own categories [Borko and Bernick, 1963]. They applied factor analysis to a correlation matrix composed of the original 90 indexing terms. An arbitary number of eigenvectors (21) were selected to represent the categories. To classify a new text, each category was given a value based upon the frequency counts and loading factors of the terms in its eigenvector. The category with the highest value was selected. This algorithm was applied to the same test set as Maron with similar results: 16 texts contained no indexing terms[2] and 48.9% of the remaining texts were correctly classified. Although the results were slightly lower than Maron's, this approach involved deriving the categories as well as classifying texts.

Hoyle [Hoyle, 1973] also developed an early text categorization system using a probabilistic approach. Given a training set of 124 that had been manually classified into 9 categories, he computed the probability of each word given a category. To classify a new text, he used Baye's rule to calculate the probability of a category given each word, summed the results, and chose the category with the highest sum. Given a test set of 124 texts, Hoyle's algorithm correctly classified 78% of them. Hoyle's work is similar to our IE-based text classification algorithms because they both use conditional probabilities to automatically derive useful indexing terms.

The AIR/X system [Fuhr *et al.*, 1991] focuses on a text classification task of a different variety. AIR/X is an approach to automated indexing where the terms, called "descriptors", come from a controlled vocabulary. The task is to index each text by labeling it with one or more of the descriptors. AIR/X uses the same conditional probability estimates as the IE-based algorithms to create mapping rules from terms to descriptors. However, AIR/X also involves additional stages, including weighting and probabilistic classification trees.

---

[2] 10 terms did not have significant loadings in any of the eigenvectors so were effectively not used.

Text routing (e.g., see papers in [Harman, 1994]) and text filtering (e.g., see papers in [CACM, 1992]) are related areas of information retrieval. Text categorization, text routing, and text filtering revolve around essentially the same task [Belkin and Croft, 1992]. In general, these systems classify documents into one or more categories. The difference between them is mainly perspective: text categorization usually involves assigning category labels to texts, text routing disseminates texts to appropriate people or groups, and text filtering removes incoming texts from a data stream if they are not relevant. All three types of systems typically involve long-term information needs and streams of incoming texts. In text categorization tasks, however, the categories are usually stable but user profiles for text routing and filtering are more dynamic (i.e., interests and users change).

An important difference between our text classification algorithms and most text routing or filtering systems is that our approach depends on an annotated training corpus. The annotated training corpus is used to construct a domain-specific dictionary that is combined with statistical techniques to classify texts. Most text routing and filtering systems, however, require only a set of texts that have been classified. These systems use domain-independent information, while our algorithms exploit a small amount of additional human effort to build a text classifier that uses domain-specific knowledge.

The traditional IR approaches (e.g., [Borko and Bernick, 1963, Fuhr *et al.*, 1991, Hoyle, 1973, Maron, 1961]) classify texts on the basis of multiple indexing terms. In contrast, the IE-based text classification algorithms do not use the entire representation of a text to decide whether it is relevant. Instead, the algorithms look for "hot spots" in a text that are highly correlated with relevance. Once a highly relevant phrase or context is found, the text is immediately classified as relevant. This approach makes sense because the indexing terms are linguistic phrases and contexts which are richer than single word terms. Since phrases and contexts are more discriminating than single words, many of them are highly correlated with relevance by themselves. By using natural language processing techniques to represent complex indexing terms, simple statistical techniques can identify reliable phrases and contexts that can be used effectively to achieve high precision.

6.3.2 AI approaches

One of the most well-known text categorization systems is CONSTRUE [Hayes and Weinstein, 1991], which achieved good results on a classification task involving 674 categories. CONSTRUE is a rule-based system built by several system developers and domain experts over the course of several years. The rule base contained sets of words and phrases to recognize important concepts. The language resembled a boolean keyword system that supported proximity and word-order relationships, approximate grammatical relations (e.g., subject-object), and weighting of individual phrases. The categorization rules allowed concept definitions to be combined with if-then relations, boolean operators, strength of the appearance weights, and text structure. Although CONSTRUE performed well, the system required over 8 person-years of effort to build. The authors claim that the lessons learned from CONSTRUE will greatly reduce the amount of time needed to produce categorization systems in other domains. However, the system relies on a rule base which must be constructed manually.

Goodman describes several experiments with rule-based and case-based approaches to text classification [Goodman, 1991]. A rule-based system combined with expectation-based parsing achieved 76% accuracy but required a large, hand-coded knowledge base of thousands of lexical and pattern definitions. A case-based system combined with expectation-based parsing achieved better results of 90% accuracy, but still relied on the hand-coded lexicon.

Masand et al. [Stanfill and Waltz, 1986] experimented with a memory-based reasoning approach to text classification. Their system used a database of 49,652 newswire articles that had been manually classified into 8 categories. To classify a new text, the K nearest neighbors were selected from the database using a standard information retrieval system. Each category was assigned a weight by summing the similarity measures of the K nearest neighbor texts; each

category that exceeded a given threshold value was assigned to the new text. This approach has the advantage that new categories can be easily added to the system just by adding new texts to the database or assigning new codes to the existing texts. The system is also domain-independent and does not require a hand-coded knowledge base. However, it does require a large database of preclassified documents. More importantly, the system achieved only moderate results. On average, the system achieved 81% recall with 70% precision, although the results varied a lot across the different categories with precision as high as 91% and as low as 53%.

IE-based text classification has the benefits of a knowledge-based approach but does not require a hand-coded knowledge base. The domain-specific dictionary can be constructed automatically by AutoSlog with only minimal effort. The annotated training corpus required for AutoSlog does require a small amount of human effort, but substantially less time that it would take to create a large knowledge base from scratch.

6.4 Information Retrieval

6.4.1 Relevance Feedback

Relevance feedback is used by some information retrieval systems to improve the performance of the system by obtaining feedback from a user (e.g., see [Buckley *et al.*, 1994, Foltz and Dumais, 1992, Haines and Croft, 1993, Harman, 1992b]). Relevance feedback techniques have consistently been shown to improve the performance of information retrieval systems. In the typical relevance feedback scenario, an IR system retrieves documents in response to a user's query and ranks them. The user then reads the most highly ranked documents and picks out the ones that are truly relevant. The relevant documents are fed back to the system for query refinement. The relevant documents may be used to add new terms to the query, to reweight the original query terms, or both.

For example, Haines and Croft [Haines and Croft, 1993] experimented with different methods for adding new query terms, including expected mutual information between term occurrence and document relevance, expected mutual information between term presence and document relevance, inverse document frequency, term frequency in relevant documents, and other variations. They also experimented with different term weighting schemes, numbers of terms added, and structured queries. Buckley et al. [Buckley *et al.*, 1994] did experiments where they expanded queries with the terms that occurred in the largest number of relevant documents and reweighted the terms using a modified Rocchio formula that assumes that all unseen documents are irrelevant.

In general, these types of techniques take advantage of manually classified documents by using general statistical measures to identify additional indexing terms and reweight the indexing terms automatically. In contrast, our text classification algorithms require that the relevant documents have also been annotated so that AutoSlog can produce an appropriate concept node dictionary for the information extraction system. General statistical techniques (conditional probabilities) are then applied to the system to identify linguistic expressions that are strongly associated with the relevant documents. Therefore our text classification algorithms require a large set of manually classified documents and additional annotations by a human whereas most relevance feedback techniques require only a small set of manually classified documents.

Relevance feedback is one way to take advantage of precategorized documents. In a sense, the relevance feedback scenario can be viewed as a classification task. The set of labeled documents supplied by the user is a training corpus. In both cases, the goal is to create a system that will correctly classify new texts that are similar to the relevant texts in the training set. One difference between text classification and relevance feedback is that relevance feedback techniques are intended to enhance the performance of a general-purpose information retrieval system. In contrast, the goal of a text classification system is to create a stand-alone text classifier. Text classification systems are intended for long-term information needs whereas relevance feedback is

aimed at short-term information needs. In practical terms, this means that text classification systems usually rely on a large training set. Sometimes the training corpus already has category labels for each text (e.g., [Masand *et al.*, 1992, Riloff and Lehnert, 1994, Riloff and Lehnert, 1992]) and sometimes it does not [Borko and Bernick, 1963, Hoyle, 1973, Maron, 1961]. However, it is not practical to obtain large training sets in the relevance feedback scenario.

Text classification systems usually require a large training set that contains both positive and negative examples of the categories. This is a reasonable requirement because the goal is to create a system for an ongoing information need. Since the classifier is a stand-alone system, the training corpus is the only source of data that it has to identify useful indexing terms. However, the information acquired from relevance feedback techniques does not need to stand on its own. The labeled documents provide additional information that is used to refine an existing system. As a result, relevance feedback techniques can benefit from a small number of labeled documents. Furthermore, most (but not all) relevant feedback system use only relevant documents whereas most text classification systems use relevant and irrelevant documents.

6.4.2 NLP and Information Retrieval

Natural language processing and information retrieval are fields that seem like they should be naturally related, but their respective communities have been relatively isolated from one another. However, there has been some work on integrating natural language processing techniques with information retrieval applications. Generally, this work falls into 2 categories: systems that use shallow, usually syntactic methods for representing phrases, and systems that use more in-depth natural language processing to capture semantics.

Syntactic methods hold promise for representing phrases in a principled manner, as opposed to methods for approximating phrase recognition used by some traditional IR systems which typically search for words in close proximity to one another. As an example of this approach, Strzalkowski [Strzalkowski, 1993] developed a system that uses a fast partial parser to improve the performance of a traditional IR system. The system uses natural language processing for stemming, word and phrase clustering, and selectional restrictions based on head-modifier structures. The phrases are also clustered on the basis of a statistical similarity measure, which helps the system handle synonymy. The linguistic processing is used both for document representation and query expansion. The system showed small but consistent performance improvements over the traditional IR system on 50 TREC queries.

In general, syntactic approaches have performed at best only slightly better than traditional word-based methods (see [Dillon, 1983, Lewis, 1992, Ruge *et al.*, 1991] for other syntactic approaches to automatic phrase indexing). These results are not surprising because most syntactic approaches operate in the same paradigm as traditional IR systems. Although phrases are represented more carefully, the syntactically-based systems still retrieve texts using statistical methods that rank documents based on accumulated evidence from individual words and phrases. However, they have the advantage that they are usually domain-independent and can be applied to any corpus.

An alternative approach is to use natural language processing to represent texts by their semantic content. For example, FERRET [Mauldin, 1989, Mauldin, 1991] is a conceptual information retrieval system that uses a sentence analyzer to parse texts and queries into conceptual dependency (CD) representations [Schank, 1975]. If the CD graph of the query matches the CD graph of a text, then the text is retrieved. Mauldin claims that the canonical representation of CD should achieve better recall than word-based systems, and that CD representations can support language-independence.[3] On a test set of 5 queries and corpus of 533 astronomy texts, FERRET

[3]For example, a query could be in Japanese and the texts in English as long as the system has sentence analyzers for both languages.

achieved 52% recall with 65% precision and a keyword system had much lower recall, 19%, but with better precision, 79% [Mauldin, 1991]. FERRET used large knowledge bases, including the original FRUMP [DeJong, 1982] script database, 5 new scripts for astronomy, a hand-coded lexicon of over 12000 frames, and rules for recognizing proper names. However, FERRET also includes a component for learning new scripts automatically [Mauldin, 1989] and uses Webster's Seventh dictionary to increase its lexicon and recognize synonyms.

NLDB [Rau and Jacobs, 1991] is a system that automatically segments text databases for information retrieval applications. NLDB uses lexical analysis, pattern matching, and company name recognition to assign keywords and topic indicators to texts. The keywords are segmented into different categories so a user can search the database more effectively. The authors did not evaluate the NLDB system as a whole, but reported results for the company name recognizer and the topic analyzer. Over several thousand stories, the company name recognizer achieved over 95% precision compared to a human and extracted 25% more company names than the human. The topic analyzer achieved over 90% precision with slightly less than 90% coverage of the topics. However, the system relies on a lexicon of 8000 root words, a concept hierarchy of 1000 concepts, and several hundred patterns for 60 topic categories. New patterns and lexical entries must be added to the lexicon for new categories.

Semantically-based natural language processing systems can perform well in limited domains, but they are not domain-independent. In general, they require large lexicons or knowledge bases for domain-specific words and concepts. The main advantage of the IE-based text classification algorithms is that they benefit from natural language processing because the information extraction system extracts semantic information but the domain-specific dictionary required for information extraction can be acquired easily and quickly. As a result, the IE-based text classification systems are knowledge-based but can be ported across domains with only a small amount of effort.

6.5 Case-Based Reasoning

The case-based text classification algorithm presented in Section 4.5 uses a case-based approach to classify texts on the basis of rich natural language contexts instead of isolated words or phrases. Although the focus of this work is not on case-based reasoning per se, it does have some contributions to the case-based reasoning community.

In particular, two aspects of this work make it different from most case-based reasoning systems. First, it uses a very large case base. The experiments described in Section 4.5.5 used a case base containing 6868 cases. Most CBR systems rely on case bases that are significantly smaller. Second, the correct classification of the cases is not known. The training corpus provides the correct classification for each text, but not for each case. Many texts contain multiple cases so it is common for a text to contain both relevant cases (i.e., cases that contain relevant information), and irrelevant cases (i.e., cases that do not contain relevant information). An irrelevant text must contain only irrelevant cases but a relevant text may contain both relevant and irrelevant cases. This is an example of the credit assignment problem. For each relevant text, we know that at least one of its cases contains relevant information but we don't know which one(s).

Most case-based reasoning systems retrieve one or a few cases that are similar to the new case and apply them directly (e.g., [Ashley, 1990, Hammond, 1986, Kolodner and Simpson, 1989]). The most obvious approach for the text classification algorithms would be to retrieve one case that is most similar to the new case. If the retrieved case is relevant then the new case would be classified as relevant; otherwise, it would be classified as irrelevant. However, this approach is not feasible because the classifications of the individual cases are not known.

Instead, the case-based algorithm retrieves many cases that are similar to the new case along specific dimensions. If a high percentage of the retrieved cases came from relevant texts then it assumes that this is not a coincidence. The retrieved cases probably share something in common that makes them all relevant. We know that they all share the index used to retrieve them.

Therefore the algorithm infers that the index probably represents information that is relevant to the domain. Since the new case also shares the index, it follows that the new case contains information that is relevant to the domain so it should be classified as relevant.

This statistical approach is one way to solve the credit assignment problem. In case-based reasoning systems, one can imagine a domain where cases are subparts of a larger structure but only the classification of the larger structure is available. In a planning domain, for example, the cases might represent subplans but only the classification of the plan as a whole is known (e.g., whether it was successful or unsuccessful). Our statistical approach would identify which cases (subplans) were most highly correlated with success or failure, even though the cases were not explicitly indexed in this fashion. However, this approach is most appropriate for domains in which a case can be judged independently from other cases (e.g., in a plan, actions may be dependent on one another).

6.6 Summary

In this chapter, we described previous research that relates to automated dictionary construction and text classification. We concluded that:

- AutoSlog is different from previous work on automated dictionary construction because it does not rely on a partial lexicon of words that are already defined. AutoSlog builds its dictionary entries completely from scratch by exploiting a training corpus of annotated texts.
- AutoSlog exemplifies a form of one-shot learning. AutoSlog is similar to explanation-based learning techniques because it builds concepts based on a single training instance, however it does not require an explicit domain theory.
- IE-based text classification is different from other knowledge-based approaches to text classification because it is portable across domains.
- Text classification is closely related to text routing and somewhat related to relevance feedback. Relevance feedback, however, is appropriate in situations with short-term information needs whereas text classification and routing are appropriate for long-term information needs. IE-based text classification also relies on a training corpus that has been annotated by a user.
- The case-based text classification algorithm represents a statistical approach to the credit assignment problem. In case-based reasoning systems, this approach could identify cases that are highly correlated with a category even though they are not explicitly indexed by the category.

CHAPTER 7

CONCLUSIONS

7.1 Research Contributions, Revisited

In Chapter 1, we made three general claims about the contributions of this work. In this chapter, we summarize the results presented in previous chapters that support each of these claims and briefly outline the implications of the claims. Finally, we wrap up by describing directions and promising avenues for future research in these areas.

The first claim of this dissertation involved the issue of automated dictionary construction for information extraction. We claimed that:

Claim #1: Dictionaries for information extraction can be constructed automatically.

In Chapter 3, we described a system called AutoSlog that automatically constructs dictionaries for information extraction. We evaluated AutoSlog in three different domains: terrorism, joint ventures, and microelectronics. In the terrorism domain, a dictionary created by AutoSlog achieved 98% of the performance of a hand-crafted dictionary. Furthermore, that hand-crafted dictionary required approximately 1500 person-hours of effort to build whereas the AutoSlog dictionary was constructed in only 5 hours.

There were no hand-crafted dictionaries for the joint ventures or microelectronics domains, but we were pleased with AutoSlog's performance in these domains. The microelectronics domain posed some difficulties for AutoSlog because it is dominated by technical phrases and jargon. In Chapter 5, we distinguished between event-based domains and technical domains. We concluded that AutoSlog is appropriate for event-based domains, but not necessarily for technical domains. However, the task must also be considered. If a task requires the identification of events and roles related to technical information, then AutoSlog may be useful in conjunction with keywords to recognize expressions that identify relationships between objects.

We also presented results from two experiments which demonstrated that people can use AutoSlog effectively with only minimal training. In the first experiment, students used AutoSlog to create their own dictionaries for terrorism, most of which obtained at least 75-85% of the performance of a hand-crafted dictionary. In the second experiment, two government analysts used AutoSlog to create their own dictionaries for the joint ventures domain. The dictionaries produced by the analysts achieved scores slightly better than a dictionary constructed by an NLP researcher. This experiment demonstrated that domain experts who have no technical background can use AutoSlog effectively.

Claim #1 contributes to the field of natural language processing in several ways. First, it demonstrates that information extraction systems can be portable across domains. In the past, the domain knowledge required for knowledge-based natural language processing systems has been manually coded. AutoSlog shows that there are automatic methods for acquiring lexical domain knowledge that are applicable in many domains.

Second, AutoSlog shows that local syntactic information is often enough to generate semantic patterns describing role relationships. AutoSlog derives concept node definitions strictly from annotated texts[1] and syntactic information provided by CIRCUS, but the resulting definitions represent linguistic patterns that predict semantic relationships between events and objects. Not all of the definitions produced by AutoSlog are semantically correct, but a small amount of human effort (5 hours in the terrorism domain, 15-20 hours in the joint ventures and microelectronics domains) can be applied to filter out the bad definitions.[2] As a result, a dictionary of semantic case frames can be acquired quickly by leveraging a fully automated method with minimal human effort.

Since this approach still depends on a manually encoded training corpus, we have not yet eliminated the knowledge engineering bottleneck for natural language processing. But we have significantly changed the nature of the bottleneck by transferring it from the hands of NLP experts to NLP novices. The knowledge engineering demands can be met by anyone familiar with the domain. Knowledge-based NLP systems will be practical for real-world applications only when their domain-dependent dictionaries can be constructed quickly. Our approach to automated dictionary construction is a significant step toward making information extraction systems scalable and portable to new domains.

The second claim of this dissertation addressed the integration of natural language processing with information retrieval:

Claim #2: Information extraction can support high precision text classification.

This claim contributes to both the natural language processing communities and the information retrieval communities by demonstrating that some information retrieval tasks can benefit from natural language processing techniques. In Chapter 4, we showed that information extraction (IE) can be used as a basis for text classification. We presented three different text classification algorithms based on information extraction: the relevancy signatures algorithm, the augmented relevancy signatures algorithm, and a case-based text classification algorithm. We referred to this general approach as IE-based text classification.

In the terrorism domain, all three IE-based text classification algorithms outperformed a word-based algorithm. Furthermore, the results showed that using more linguistic information improves performance. In general, relevancy signatures performed better than words, augmented relevancy signatures performed better than relevancy signatures, and the case-based algorithm performed better than augmented relevancy signatures.

In the joint ventures domain, the IE-based algorithms also performed better than a word-based algorithm but the differences between the three IE-based algorithms were less dramatic. However, the word-based algorithm outperformed the relevancy signatures algorithm in the microelectronics domain.

Chapter 5 discusses general properties of domains and tasks that determine the effectiveness of the different approaches. We concluded that technical domains are more well-suited for keyword-based techniques but event-based domains are more appropriate for IE-based techniques. However, the classification task is a crucial factor in determining which algorithms are most appropriate. In general, keywords are sufficient to handle technical information and general event references but relevancy signatures are more adept at recognizing specific event references and role objects. Augmented relevancy signatures provide the ability to recognize semantic properties of role objects and the case-based algorithm is capable of implicitly recognizing some relationships involving multiple role objects.

---

[1] Or texts and associated answer keys

[2] The difference in filtering times across the domains was primarily due to the amount of input data and the generalization modules used for in the JV and ME domains (see Section 3.5.2.4).

There have been previous attempts to integrate NLP with IR, but few have demonstrated significant performance improvements over traditional IR techniques. Our contribution has been to demonstrate that IE-based text classification can obtain high levels of precision, at significant recall levels, that are difficult to achieve with word-based techniques. Furthermore, in the terrorism domain, we have shown that using increasing amounts of linguistic information can show dramatic performance improvements.

The third claim combines the two major themes of this dissertation: automated dictionary construction and IE-based text classification.

> *Claim #3: Knowledge-based text classification systems can be portable across domains.*

As we explained in Chapter 1, the big picture outlined in this dissertation is a portable, knowledge-based text classification system. By combining AutoSlog with the text classification algorithms, a knowledge-based text classification system can be brought up for a new domain with only a small amount of human effort. The input required for the system is an annotated training corpus of relevant and irrelevant texts, where the domain-specific information in the relevant texts has been marked. Given this training corpus, AutoSlog can automatically build a concept node dictionary for the domain. A few hours of human-assisted filtering ensures that the dictionary is reliable.

The text classification algorithms use the training corpus a second time to determine which types of domain-specific information are reliable for classification purposes. The text classification algorithms are fully automated and domain-independent. The result is a knowledge-based text classifier for a specific domain.

Knowledge-based systems have been successful with many tasks [Carbonell, 1979a, Cullingford, 1978, DeJong, 1982, Hayes and Weinstein, 1991, Lehnert, 1991, Mauldin, 1991] but, in the past, they have relied on manually constructed knowledge bases. The combination of AutoSlog and the corpus-based text classification algorithms allows domain-specific classifiers to be constructed quickly. This corpus-based approach to automated knowledge acquisition exploits a small amount of human effort to create a large, domain-specific systems.

This scheme represents a new approach for developing domain-specific systems without hand-coding domain knowledge. This approach opens up possibilities for automatically building domain-specific systems for a variety of tasks. In the next section, we discuss some possible directions for future research in the areas of automated dictionary construction, text processing, and information retrieval.

7.2 Future Research

7.2.1 Automated Dictionary Construction

AutoSlog has demonstrated that it can generate good concept node definitions but it also produces bad concept nodes along with the good ones. One possible research direction is to try to reduce the number of bad definitions. Although some improvements along these lines are certainly possible, it is unlikely that the bad definitions can be suppressed without losing many good ones as well, or without invoking external knowledge. An alternative approach is to apply a feedback mechanism to evaluate the utility of a proposed definition. For example, one could apply the proposed concept node to the training corpus and gather statistics to evaluate its performance. Concept nodes that performed poorly would be discarded automatically. Concept nodes that performed well would be passed along for subsequent filtering by a human, as before. This could substantially reduce the time required for the human-in-the-loop. If the feedback mechanism is reliable enough then the human-in-the-loop could be bypassed altogether and the statistics alone could be used to automatically filter the dictionary.

However, evaluating the performance of the concept nodes is a non-trivial task. There are several issues that make this problem more difficult than it might seem, which are explained in a footnote in Section 3.5.2.5. Nevertheless, it is a promising avenue for future research. Furthermore, automatically filtered dictionaries have the potential to achieve better performance than human-filtered dictionaries. It is possible that the people-in-the-loop who participated in the AutoSlog experiments discarded definitions that were, in fact, reliable and useful. Or they may have kept definitions that were not reliable even though they seemed to be useful intuitively. It is often difficult for humans to know which patterns will be the most effective; a statistical feedback mechanism could report the actual performance of the definitions on real data to predict how effective they will be in practice.

7.2.2 Text Segmentation

Discourse analysis is one of the most challenging problems in natural language processing. Although the term "discourse analysis" often refers to processing conversational dialogues, it is also used to refer to intersentential problems in text processing. For example, in the recent message understanding conferences, most of the systems were relatively good at extracting information from individual sentences. However, there are many issues involved with extracting and combining information across sentences, such as co-reference resolution (i.e., anaphor resolution), event segmentation, topic identification, etc. These problems caused a great deal of difficulty for the MUC participants [Iwanska *et al.*, 1991]. Discourse analysis is an important problem in many text processing applications but is a poorly understood area that deserves future attention.

The text classification algorithms that we presented could be adapted to handle one of the most important problems in discourse analysis: text segmentation. For example, the information extraction task for terrorism specified that the information corresponding to each event had to be put into a different output template. Many of the texts in the MUC-4 corpus described multiple terrorist incidents, for example two bombings and one kidnapping. So the MUC-4 systems had to determine how many separate events are mentioned in a text, and which pieces of information correspond to each event. This is essentially a text segmentation problem; that is, the text needs to be segmented into pieces corresponding to the separate events. One way to do this would be to use the text classification algorithms to classify each sentence with respect to an event type. For example, the first three sentences might be classified as bombing sentences and the next four sentences might be classified as kidnapping sentences. Adjacent sentences with similar event types would be assigned to the same text segment. The case-based text classification algorithm provides the richest sentence representations for comparing adjacent sentences. Although text segmentation is a difficult problem, IE-based text classification holds promise for attacking this problem with a combination of statistics and semantics.

7.2.3 Text Summarization

The goal of IR systems is to retrieve documents that are believed to be of interest to a user. The documents are usually ranked so that the documents most likely to be relevant are listed first. However, the user generally has to read or skim each document to determine whether it is actually of interest to them. At best, traditional IR systems can highlight the terms that caused the document to be highly ranked. In many applications, it would be useful if the system could automatically generate a summary of the text for the user to review.

The IE-based text classification algorithms are well-suited for text summarization. In particular, the case-based text classification algorithm is useful for identifying sentences that are strongly associated with relevance. Presumably, these sentences are the ones that contain the most important information with respect to the topic of interest. The same sentences that are judged to be most reliable for text classification could be used to generate a summary of the

text. A short summary could be generated by using only the most highly relevant sentences; alternatively, a longer summary could be generated by using all of the sentences that are positively associated with the domain. These summaries would save the user from having to read the entire document. Also, these summaries would be domain-specific (i.e., topic specific) so they would include only the information in the text that corresponds to the user's interest. Since many texts are long and cover multiple topics, domain-specific summaries could substantially reduce the time required for the user to find the relevant information and determine whether it is of interest.

7.2.4 Multi-Class Text Categorization

The text classification experiments presented so far involved a binary classification task. One extension to this work is to apply the text classification algorithms to multi-class categorization problems. For example, the binary classification task in the terrorism domain requires the ability to separate the relevant texts from the irrelevant texts. A more ambitious task is multi-class categorization. For example, in the terrorism domain a categorization system might classify texts based on event type to separate bombings, kidnappings, murders, etc. Each text would be assigned one or more event categories depending upon what events are found in the text.

Preliminary investigations suggest that the best approach to multi-class categorization is to generate relevancy signatures for each event type separately. For example, bombing relevancy signatures could be generated by determining which signatures are most highly correlated with texts that have bombing templates in the corpus.[3] A new text would then be assigned to the bombing category if it contains any of the highly ranked bombing signatures.

An alternative approach is to collect all of the signatures produced by a text and gather the event types represented by these signatures. Categories could be assigned by choosing the event types most frequently represented by the signatures. However, this approach leads to some difficulties. First, low frequency event types are at a disadvantage. For example, the MUC-4 corpus contains a lot of bombings but very few robberies. Therefore, a single highly relevant robbery signature warrants a robbery classification even if the text also contains many bombing signatures. Treating the categories uniformly makes it difficult to assign low frequency categories while maintaining good precision on high frequency categories. We concluded that multiple categories are more naturally and effectively handled by deriving individual signatures and threshold values for each category.

7.2.5 Information Retrieval

Most information retrieval tasks focus on short-term information needs, which implies that the system must be domain-independent. However, both AutoSlog and the text classification algorithms could be integrated into traditional IR systems. Although AutoSlog creates domain-specific dictionaries, there are many possible queries for each domain. For example, one could construct hundreds of different queries for retrieving terrorism texts. If a domain is common enough, then an information retrieval system might benefit from AutoSlog's dictionary of domain patterns. AutoSlog's patterns could be used to recognize important concepts in a query, including expressions that might be difficult for a word-based system to recognize. Or AutoSlog's patterns could be used for query expansion to help in identifying synonymous terms. For large databases, it may not be practical to apply to CIRCUS to every text. However, AutoSlog's patterns could be approximated using regular expressions. It is not clear how well the approximations would work, but the practical advantages of regular expressions might be worth sacrificing some linguistic functionalities.

---

[3]Or bombing annotations in an annotated corpus.

As we described in Section 6.4.1, relevance feedback is similar to classification in the sense that both tasks exploit a set of preclassified texts to retrieve or classify new texts. In the relevance feedback scenario, a user typically identifies a small set of relevant texts that are then fed back to the system to improve its performance. In contrast, the text classification algorithms rely on a large set of training texts that are used once to generate useful indexing terms. These approaches could be combined for IR tasks with long-term information needs, for example in text filtering environments. Initially, a user might provide a query which would be used by a traditional IR system to filter out irrelevant documents (e.g., only the 20 mostly highly ranked documents would be shown to the user). The user could then provide feedback which is used to update the query. Over time, the set of manually classified documents would be large enough to apply the text classification algorithms. Whenever the text classification algorithms were able to identify a useful indexing term, this term could also be used to recognize relevant documents, either by itself or as another term in the query. However, the text classification algorithms do rely on AutoSlog so the user must be willing to annotate the relevant texts as well as classify them.

7.3 Summary

In summary, we have presented a new model for text classification that uses an underlying information extraction system to achieve high-precision text classification. Information extraction techniques provide the system with the strengths of natural language processing without the difficulties of in-depth text understanding. As a practical matter, our text classification algorithms and the information extraction system can be easily ported to new domains. We believe that shallow natural language processing techniques, such as information extraction, can be used effectively to support many information retrieval applications. By pipelining traditional IR techniques with natural language processing capabilities, we may be able to achieve better performance than either technology could support on its own.

APPENDIX A

CONCEPT NODE SUBTYPES FOR JOINT VENTURES

Entity Name Relationship Types:
- jv-child
- jv-parent
- jv
- parent
- none

Entity Name Subtypes:
- company
- government
- person
- none

Facility Name Subtypes:
- communications
- factory
- farm
- office
- mine
- site
- store
- transportation
- utilities
- warehouse
- none

Product/Service Subtypes:
- finance
- production
- research
- sales
- service
- none

APPENDIX B

SEMANTIC FEATURES FOR JOINT VENTURES

Semantic Feature	**# Words**	**Examples**
animal	3	cattle, dogs, kangaroo
building	31	condominium, office, plant
company	1004	Aeroflot, Corp, K-Swiss
chemical	55	acid, drugs, glycol
entity	156	association, Baxter, consortium
finance	24	banking, investment, securities
food	35	coffee, pizza, rice
location	48	Boston, France, province
material-good	377	batteries, machine, movies
natural-resource	24	coal, silver, water
person	453	Bergen, Costello, Fitzgerald
research	12	study, methods, tests
service	74	advertising, charters, law
site	26	canyon, estate, parkway
store	32	Bloomingdales, hotel, restaurant
thing	299	activity, combination, joint-venture
vehicle	73	airplanes, corolla, sub-compacts
TOTAL	2726	

APPENDIX C

COLLOCATION FREQUENCIES FOR PREPOSITIONAL PHRASE ATTACHMENT

Collocation Frequencies in the Joint Ventures Domain[4]

"with"		"by"		"for"		"at"	
comma	279	owned	91	*comma*	149	*start*	109
venture	270	up	62	*start*	131	capitalized	85
agreement	86	pct	60	market	64	aimed	60
start	84	*start*	53	demand	48	*comma*	54
talks	57	percent	38	plans	41	closed	40
compared	51	made	35	calls	35	close	25
up	46	led	30	responsible	30	estimated	22
deal	34	year	30	spokesman	29	plant	19
ventures	26	*comma*	26	called	26	year	16
ties	25	-owned	22	call	26	valued	16
company	24	supplied	22	yen	24	production	13
and	24	operated	19	way	20	look	13
negotiations	24	headed	19	contract	20	and	12
contract	22	held	16	million	19	produced	11
jointly	22	provided	15	account	19	workers	11
along	21	and	15	japan	19	analyst	11
agreed	21	developed	14	venture	19	closing	10
analyst	21	run	14	looking	18	official	10
relations	19	used	14	calling	18	built	9
negotiating	18	approval	13	accounts	18	looking	9

[4]The italicized words represent special tokens for punctuation; *start* is a special token that denotes the beginning of a sentence. When resolving attachments, only nouns and verbs are considered as possible attachment points so high collocation frequencies of other words and special tokens do not affect the algorithm.

Collocation Frequencies in the Microelectronics Domain

"with"		"by"		"for"		"at"	
comma	244	*start*	71	*start*	120	*start*	85
start	81	developed	46	*comma*	109	priced	53
compared	40	used	38	used	79	*comma*	37
agreement	39	*comma*	25	manager	57	aimed	32
compatible	34	produced	23	need	51	researchers	26
along	33	made	23	designed	44	manager	25
working	30	led	18	market	44	marketing	21
work	26	report	15	technology	39	looking	20
up	22	acquired	14	system	29	look	17
devices	18	formed	13	process	28	installed	14
used	17	provided	13	million	27	targeted	13
equipped	17	and	10	*rparen*	25	analyst	11
deal	16	driven	9	applications	24	system	11
conjunction	15	built	9	demand	23	center	10
use	15	achieved	9	available	23	operate	10
associated	14	followed	9	orders	22	products	9
made	14	caused	9	equipment	22	is	9
system	13	production	9	looking	21	operating	9
chips	13	dominated	9	account	18	running	9
fabricated	12	capacity	8	*lparen*	17	and	9

BIBLIOGRAPHY

[Ashley, 1990] Ashley, K. *Modelling Legal Argument: Reasoning with Cases and Hypotheticals.* The MIT Press, Cambridge, MA, 1990.

[Belkin and Croft, 1992] Belkin, Nicholas and Croft, W. Bruce. Information Filtering and Information Retrieval: Two Sides of the Same Coin? *Communications of the ACM*, 35(12):29–38, 1992.

[Borko and Bernick, 1963] Borko, H. and Bernick, M. Automatic Document Classification. *J. ACM*, 10(2):151–162, 1963.

[Buckley *et al.*, 1994] Buckley, Chris, Salton, Gerard, and Allan, James. The Effect of Adding Relevance Information in a Relevance Feedback Environment. In *Proceedings, SIGIR 1994*, pages 292–300, 1994.

[CACM, 1992] Communications of the ACM, December 1992.

[Callan *et al.*, 1992] Callan, J. P., Croft, W. B., and Harding, S. M. The INQUERY Retrieval System. In *Proceedings of the Third International Conference on Database and Expert Systems Applications*, pages 78–83, 1992.

[Carbonell, 1979a] Carbonell, J. G. *Subjective Understanding: Computer Models of Belief Systems.* PhD thesis, Research Report 150, Computer Science Department, Yale University, 1979.

[Carbonell, 1979b] Carbonell, J. G. Towards a Self-Extending Parser. In *Proceedings of the 17th Meeting of the Association for Computational Linguistics*, pages 3–7, 1979.

[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*, pages 798–803. AAAI Press/The MIT Press, 1993.

[Church, 1989] Church, K. A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. In *Proceedings of the Second Conference on Applied Natural Language Processing*, 1989.

[Croft *et al.*, 1991] Croft, W. B., Turtle, H. R., and Lewis, D. D. The Use of Phrases and Structured Queries in Information Retrieval. In *Proceedings, SIGIR 1991*, pages 32–45, 1991.

[Crouch and Yang, 1992] Crouch, Carolyn J. and Yang, Bokyung. Experiments in Automatic Statistical Thesaurus Construction. In *Proceedings, SIGIR 1992*, pages 77–88, 1992.

[Crouch, 1988] Crouch, Carolyn J. A Cluster-Based Approach to Thesaurus Construction. In *Proceedings of the Eleventh International Conference on Research and Development in Information Retrieval*, pages 309–320, 1988.

[Cullingford, 1978] Cullingford, R. E. *Script Application: Computer Understanding of Newspaper Stories.* PhD thesis, Research Report 116, Computer Science Department, Yale University, 1978.

[DeJong and Mooney, 1986] DeJong, Gerald and Mooney, R. Explanation-Based Learning: An Alternative View. *Machine Learning*, 1:145–176, 1986.

[DeJong, 1982] DeJong, Gerald. An Overview of the FRUMP System. In Lehnert, W. and Ringle, M., editors, *Strategies for Natural Language Processing*, pages 149–177. Lawrence Erlbaum Associates, 1982.

[Dillon, 1983] Dillon, M. FASIT: A Fully Automatic Syntactically Based Indexing System. *Journal of the American Society for Information Science*, 34(2):99–108, 1983.

[Dolan *et al.*, 1993] Dolan, William, Vanderwende, Lucy, and Richardson, Stephen D. Automatically Deriving Structured Knowledge Bases from On-Line Dictionaries. In *Proceedings of the First Conference of the Pacific Association for Computational Linguistics*, pages 5–14, 1993.

[Fagan, 1989] Fagan, J. The Effectiveness of a Nonsyntactic Approach to Automatic Phrase Indexing for Document Retrieval. *Journal of the American Society for Information Science*, 40(2):115–132, 1989.

[Fikes *et al.*, 1972] Fikes, R. E., Hart, P. E., and Nilsson, N. J. Learning and Executing Generalized Robot Plans. *Artificial Intelligence*, 3:251–288, 1972.

[Fisher, 1987] Fisher, D. H. Knowledge Acquisition Via Incremental Conceptual Clustering. *Machine Learning*, 2:139–172, 1987.

[Foltz and Dumais, 1992] Foltz, Peter W. and Dumais, Susan T. Personalized Information Delivery: An Analysis of Information Filtering Methods. *Communications of the ACM*, 35(12):51–60, 1992.

[Frakes and Baeza-Yates, 1992] Frakes, William B. and Baeza-Yates, Ricardo, editors. *Information Retrieval: Data Structures and Algorithms*. Prentice Hall, Englewood Cliffs, NJ, 1992.

[Francis and Kucera, 1982] Francis, W. and Kucera, H. *Frequency Analysis of English Usage*. Houghton Mifflin, Boston, MA, 1982.

[Fuhr *et al.*, 1991] Fuhr, N., Hartmann, S., Lustig, G., Schwantner, M., and Tzeras, Konstadinos. AIR/X - A Rule-Based Multistage Indexing System for Large Subject Fields. In *Proceedings of RIAO 91*, pages 606–623, 1991.

[Goodman, 1991] Goodman, M. Prism: A Case-Based Telex Classifier. In *Proceedings of the Second Annual Conference on Innovative Applications of Artificial Intelligence*, pages 25–37. AAAI Press, 1991.

[Granger, 1977] Granger, R. H. FOUL-UP: A Program that Figures Out Meanings of Words from Context. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 172–178, 1977.

[Haines and Croft, 1993] Haines, David and Croft, Bruce. Relevance Feedback and Inference Networks. Computer science technical report 93-31, University of Massachusetts, Amherst, MA, 1993.

[Hammond, 1986] Hammond, K. CHEF: A Model of Case-Based Planning. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 267–271. Morgan Kaufmann, 1986.

[Harman, 1992a] Harman, D. The DARPA Tipster Project. *SIGIR Forum*, 26(2):26–28, 1992.

[Harman, 1992b] Harman, Donna. Relevance Feedback and Other Query Modification Techniques. In *Information Retrieval: Data Structures and Algorithms*, chapter 11, pages 241–263. Prentice Hall, 1992.

[Harman, 1993] Harman, D., editor. *The First Text REtrieval Conference (TREC1).* National Institute of Standards and Technology Special Publication 200-207, Gaithersburg, MD, 1993.

[Harman, 1994] Harman, D., editor. *The Second Text REtrieval Conference (TREC2).* National Institute of Standards and Technology Special Publication 500-215, Gaithersburg, MD, 1994.

[Hayes and Weinstein, 1991] Hayes, Philip J. and Weinstein, Steven P. Construe-TIS: A System for Content-Based Indexing of a Database of News Stories. In *Proceedings of the Second Annual Conference on Innovative Applications of Artificial Intelligence*, pages 49–64. AAAI Press, 1991.

[Hobbs *et al.*, 1992] Hobbs, Jerry R., Appelt, Douglas, Tyson, Mabry, Bear, John, and Israel, David. SRI International: Description of the FASTUS System Used for MUC-4. In *Proceedings of the Fourth Message Understanding Conference (MUC-4)*, pages 268–275, San Mateo, CA, 1992. Morgan Kaufmann.

[Hoyle, 1973] Hoyle, W. Automatic Indexing and Generation of Classification Systems by Algorithm. *Information Storage and Retrieval*, 9(4):233–242, 1973.

[Iwanska *et al.*, 1991] Iwanska, Lucja, Appelt, Douglas, Ayuso, Damaris, Dahlgren, Kathy, Glover Stalls, Bonnie, Grishman, Ralph, Krupka, George, Montgomery, Christine, and Riloff, Ellen. Computational Aspects of Discourse in the Context of MUC-3. In *Proceedings of the Third Message Understanding Conference (MUC-3)*, pages 256–282, San Mateo, CA, 1991. Morgan Kaufmann.

[Jacobs and Zernik, 1988] Jacobs, P. and Zernik, U. Acquiring Lexical Knowledge from Text: A Case Study. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 739–744, 1988.

[Jacobs *et al.*, 1991] Jacobs, Paul S., Krupka, George R., and Rau, Lisa F. Lexico-Semantic Pattern Matching as a Companion to Parsing in Text Understanding. In *Proceedings of the Fourth DARPA Speech and Natural Language Workshop*, pages 337–342. Morgan Kaufmann, 1991.

[Kim and Moldovan, 1993] Kim, J. and Moldovan, D. Acquisition of Semantic Patterns for Information Extraction from Corpora. In *Proceedings of the Ninth IEEE Conference on Artificial Intelligence for Applications*, pages 171–176, Los Alamitos, CA, 1993. IEEE Computer Society Press.

[Kolodner and Simpson, 1989] Kolodner, J. and Simpson, R. The MEDIATOR: Analysis of an Early Case-Based Problem Solver. *Cognitive Science*, 13(4):507–549, 1989.

[Krovetz and Croft, 1989] Krovetz, R. and Croft, W. B. Word Sense Disambiguation Using Machine-Readable Dictionaries. In *Proceedings, SIGIR 1989*, 1989.

[Lehnert and Sundheim, 1991] Lehnert, W. G. and Sundheim, B. A Performance Evaluation of Text Analysis Technologies. *AI Magazine*, 12(3):81–94, 1991.

[Lehnert *et al.*, 1992a] Lehnert, W., Cardie, C., Fisher, D., McCarthy, J., Riloff, E., and Soderland, S. University of Massachusetts: Description of the CIRCUS System as Used for MUC-4. In *Proceedings of the Fourth Message Understanding Conference (MUC-4)*, pages 282–288, San Mateo, CA, 1992. Morgan Kaufmann.

[Lehnert *et al.*, 1992b] Lehnert, W., Cardie, C., Fisher, D., McCarthy, J., Riloff, E., and Soderland, S. University of Massachusetts: MUC-4 Test Results and Analysis. In *Proceedings of the Fourth Message Understanding Conference (MUC-4)*, pages 151–158, San Mateo, CA, 1992. Morgan Kaufmann.

[Lehnert *et al.*, 1993a] Lehnert, W., McCarthy, J., Soderland, S., Riloff, E., Cardie, C., Peterson, J., Feng, F., Dolan, C., and Goldman, S. UMass/Hughes: Description of the CIRCUS System as Used for MUC-5. In *Proceedings of the Fifth Message Understanding Conference (MUC-5)*, pages 277–291, San Francisco, CA, 1993. Morgan Kaufmann.

[Lehnert *et al.*, 1993b] Lehnert, W., McCarthy, J., Soderland, S., Riloff, E., Cardie, C., Peterson, J., Feng, F., Dolan, C., and Goldman, S. UMass/Hughes: Description of the CIRCUS System Used for TIPSTER Text Extraction. In *Proceedings of the TIPSTER Text Program (Phase I)*, pages 241–256, San Francisco, CA, 1993. Morgan Kaufmann.

[Lehnert, 1991] Lehnert, W. Symbolic/Subsymbolic Sentence Analysis: Exploiting the Best of Two Worlds. In Barnden, J. and Pollack, J., editors, *Advances in Connectionist and Neural Computation Theory, Vol. 1*, pages 135–164. Ablex Publishers, Norwood, NJ, 1991.

[Lewis, 1992] Lewis, David Dolan. *Representation and Learning in Information Retrieval.* PhD thesis, Computer Science Department, University of Massachusetts, Amherst, MA 01003. Technical Report 91-93., 1992.

[Liddy *et al.*, 1993] Liddy, Elizabeth D., Paik, Woojin, and Yu, S. Edmund. Document Filtering Using Semantic Information from a Machine Readable Dictionary. In *Proceedings of the Workshop on Very Large Corpora: Academic and Industrial Perspectives*, pages 20–29, 1993.

[Marcus *et al.*, 1993] Marcus, M., Santorini, B., and Marcinkiewicz, M. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[Maron, 1961] Maron, M. Automatic Indexing: An Experimental Inquiry. *J. ACM*, 8:404–417, 1961.

[Masand *et al.*, 1992] Masand, Brij, Linoff, Gordon, and Waltz, David. Classifying News Stories Using Memory Based Reasoning. In *Proceedings, SIGIR 1992*, pages 59–65, 1992.

[Mauldin, 1989] Mauldin, M. *Information Retrieval by Text Skimming.* PhD thesis, School of Computer Science, Carnegie Mellon University, 1989.

[Mauldin, 1991] Mauldin, M. Retrieval Performance in FERRET: A Conceptual Information Retrieval System. In *Proceedings, SIGIR 1991*, pages 347–355, 1991.

[Mitchell *et al.*, 1986] Mitchell, T. M., Keller, R., and Kedar-Cabelli, S. Explanation-Based Generalization: A Unifying View. *Machine Learning*, 1:47–80, 1986.

[Montemagni and Vanderwende, 1992] Montemagni, S. and Vanderwende, L. Structural Patterns vs. String Patterns for Extracting Semantic Information from Dictionaries. In *Proceedings of the Fourteenth International Conference on Computational Linguistics (COLING-92)*, pages 546–552, 1992.

[MUC-3 Proceedings, 1991] *Proceedings of the Third Message Understanding Conference (MUC-3)*, San Mateo, CA, 1991. Morgan Kaufmann.

[MUC-4 Proceedings, 1992] *Proceedings of the Fourth Message Understanding Conference (MUC-4)*, San Mateo, CA, 1992. Morgan Kaufmann.

[MUC-5 Proceedings, 1993] *Proceedings of the Fifth Message Understanding Conference (MUC-5)*, San Francisco, CA, 1993. Morgan Kaufmann.

[Quinlan, 1986] Quinlan, J. R. Induction of Decision Trees. *Machine Learning*, 1:80–106, 1986.

[Rau and Jacobs, 1991] Rau, Lisa F. and Jacobs, Paul S. Creating Segmented Databases From Free Text for Text Retrieval. In *Proceedings, SIGIR 1991*, pages 337–346, 1991.

[Riesbeck, 1978] Riesbeck, C. An Expectation-Driven Production System for Natural Language Understanding. In Waterman, D. A. and Hayes-Roth, F., editors, *Pattern-directed Inference Systems*. Academic Press, 1978.

[Riloff and Lehnert, 1992] Riloff, E. and Lehnert, W. Classifying Texts Using Relevancy Signatures. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 329–334. AAAI Press/The MIT Press, 1992.

[Riloff and Lehnert, 1993] Riloff, E. and Lehnert, W. Automated Dictionary Construction for Information Extraction from Text. In *Proceedings of the Ninth IEEE Conference on Artificial Intelligence for Applications*, pages 93–99, Los Alamitos, CA, 1993. IEEE Computer Society Press.

[Riloff and Lehnert, 1994] Riloff, E. and Lehnert, W. Information Extraction as a Basis for High-Precision Text Classification. *ACM Transactions on Information Systems*, 12(3):296–333, July 1994.

[Riloff, 1993a] Riloff, E. Automatically Constructing a Dictionary for Information Extraction Tasks. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 811–816. AAAI Press/The MIT Press, 1993.

[Riloff, 1993b] Riloff, E. Using Cases to Represent Context for Text Classification. In *Proceedings of the Second International Conference on Information and Knowledge Management (CIKM-93)*, pages 105–113, New York, NY, 1993. ACM Press.

[Ruge *et al.*, 1991] Ruge, Gerda, Schwarz, Christoph, and Warner, Amy J. Effectiveness and Efficiency in Natural Language Processing for Large Amounts of Text. *Journal of the American Society for Information Science*, 42(6):450–456, 1991.

[Salton, 1971] Salton, G., editor. *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice Hall, Englewood Cliffs, NJ, 1971.

[Salton, 1989] Salton, G. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, Reading, MA, 1989.

[Schank, 1975] Schank, Roger C. *Conceptual Information Processing*, chapter 3, pages 22–82. North Holland Publishers, 1975.

[Stanfill and Waltz, 1986] Stanfill, C. and Waltz, D. Toward Memory-Based Reasoning. *Communications of the ACM*, 29(12):1213–1228, 1986.

[Strzalkowski, 1993] Strzalkowski, Tomek. Robust Text Processing in Automated Information Retrieval. In *Proceedings of the Workshop on Very Large Corpora: Academic and Industrial Perspectives*, pages 9–19, 1993.

[Tipster Proceedings, 1993] *Proceedings of the TIPSTER Text Program (Phase I)*, San Francisco, CA, 1993. Morgan Kaufmann.

[Turtle and Croft, 1991] Turtle, Howard and Croft, W. Bruce. Efficient Probabilistic Inference for Text Retrieval. In *Proceedings of RIAO 91*, pages 644–661, 1991.

[Utgoff, 1988] Utgoff, P. ID5: An Incremental ID3. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 107–120, 1988.

[Weischedel *et al.*, 1993] Weischedel, R., Meteer, M., Schwartz, R., Ramshaw, L., and Palmucci, J. Coping with Ambiguity and Unknown Words through Probabilistic Models. *Computational Linguistics*, 19(2):359–382, 1993.