Information Extraction as a Basis for High-Precision Text Classification

ELLEN RILOFF and WENDY LEHNERT
University of Massachusetts

We describe an approach to text classification that represents a compromise between traditional word-based techniques and in-depth natural language processing. Our approach uses a natural language processing task called "information extraction" as a basis for high-precision text classification. We present three algorithms that use varying amounts of extracted information to classify texts. The *relevancy signatures algorithm* uses linguistic phrases; the *augmented relevancy signatures algorithm* uses phrases and local context; and the *case-based text classification algorithm* uses larger pieces of context. Relevant phrases and contexts are acquired automatically using a training corpus We evaluate the algorithms on the basis of two test sets from the MUC-4 corpus All three algorithms achieved high precision on both test sets, with the augmented relevancy signatures algorithm and the case-based algorithm reaching 100% precision with over 60% recall on one set Additionally, we compare the algorithms on a larger collection of 1700 texts and describe an automated method for empirically deriving appropriate threshold values. The results suggest that information extraction techniques can support high-precision text classification and, in general, that using more extracted information improves performance. As a practical matter, we also explain how the text classification system can be easily ported across domains.

Categories and Subject Descriptors: H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; I.2.7 [**Artificial Intelligence**]: Natural Language Processing

General Terms: Algorithms, Experimentation, Performance

Additional Key Words and Phrases: Information extraction, text classification

1. INTRODUCTION

Traditional approaches to information retrieval use keyword searches and statistical techniques to retrieve relevant documents (e.g., Salton [1989] and Turtle and Croft [1991]). Statistical techniques take advantage of large document collections to identify words that are useful indexing terms automatically. These techniques are popular because they can be fully automated

This research was funded by NSF grant EEC-9209623, supporting the Center for Intelligent Information Retrieval at the University of Massachusetts.
Authors' address: Department of Computer Science, University of Massachusetts, Amherst, MA 01003; email: {riloff; lehnert}@cs umass.edu.

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 [Mauldin 1991].

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 generally 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.

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 whole text. For example, the sentence "an armed man took the money and fled" refers to a robbery. Together, these words clearly describe a robbery, even though none of the words is a good indexing term by itself.

Synonymy is a well-known limitation of word-based techniques because it 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., 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 (e.g., Dillon [1983], Fagan [1989], and Croft et al. [1991]). 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 great deal of work recently on knowledge-based information retrieval systems (e.g., Goodman [1991], Hayes and Weinstein [1991], Mauldin [1991], and 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 knowl-

edge engineering is a time-consuming and tedious process that may require several 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], and Hoyle [1973] for traditional IR approaches and Goodman [1991], Hayes and Weinstein [1991], and 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 predefined, 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 the context surrounding the phrases are recognized easily and handled naturally. Additionally, the system can classify texts that would be inaccessible to 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 still benefiting from a knowledge-based approach.

Finally, the emphasis of our 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 were 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 with good accuracy. Liberal algorithms are more eager to classify texts as relevant but may misclassify many texts as a result. 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 Section 2, we give a brief introduction to information extraction, the CIRCUS sentence analyzer, and the MUC-4 task and corpus on which our experiments are based. Section 3 presents three text classification algorithms that are based on an underlying information extraction system. These algorithms are distinguished by the amount of extracted information that they use to classify texts. The *relevancy signatures algorithm* uses linguistic expressions that represent phrases; the *augmented relevancy signatures algorithm* uses linguistic expressions in combination with local extracted information; and the *case-based text classification algorithm* uses multiple pieces of extracted information to represent larger natural language contexts. We evaluate each of the algorithms on the basis of two blind test sets from the MUC-4 corpus. Section 4 describes a more comprehensive set of experiments to compare the algorithms on a larger test set. We also present an automated method for deriving empirically an appropriate set of threshold values for each algorithm. Section 5 briefly describes what is needed to port the text classification system across domains, and Section 6 concludes with a discussion of the novel aspects of this work.

2. INFORMATION EXTRACTION

When considering a technology to support high-precision text classification, natural language processing (NLP) is one of the first things that comes to mind. It seems reasonable to believe that we could produce accurate classifications if we could actually *understand* the documents. However, in-depth natural language processing is an expensive endeavor that can strain computational resources. And 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 a document, 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 a document. For example, in the domain of terrorism, an information extraction system might extract the names of all perpetrators, victims, physical targets, and weapons 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 and ambiguity resolution). As a result, information extraction is a practical and feasible technology that has achieved success in the last few years [Lehnert and Sundheim 1991; MUC-3 Proceedings 1991; MUC-4 Proceedings 1992; MUC-5 Proceedings 1993].

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 structures 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, our dictionary contains two concept nodes that are 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 only 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 will produce 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 that the perpetrator is the object of the preposition "by." Figure 1 shows the concept node definitions for $murder-active$ and $murder-passive$.

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, but the soft constraints act only as preferences for fillers. Therefore, a slot may be filled even if a soft constraint is violated. Figure 2 shows a sample sentence and the resulting instantiated concept node produced by CIRCUS.

Name:	$MURDER-ACTIVE$	$MURDER-PASSIVE$
Trigger Word:	murdered	murdered
Variable Slots:	((perpetrator (*SUBJECT* 1)) (victim (*DOBJ* 1)))	((victim (*SUBJECT* 1)) (perpetrator (*PP* (is-prep? '(by)))))
Slot Constraints:	((class perpetrator *SUBJECT*) (class victim *DOBJ*))	((class victim *SUBJECT*) (class perpetrator *PP*))
Constant Slots:	(type murder)	(type murder)
Enabling Conditions:	((active))	((passive))

Fig. 1. Two concept node definitions.

Sentence: Three peasants were murdered by guerrillas.

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

Fig. 2. An instantiated concept node.

Since concept nodes are the only form of CIRCUS output, a good dictionary of concept nodes is crucial for effective information extraction. The UMass/MUC-4 system [Lehnert et al. 1992] 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. In Section 5 we will explain how these concept node definitions can be acquired automatically [Riloff 1993a; Riloff and Lehnert 1993a].

2.2 The MUC-4 Task and Corpus

Our interest in information extraction was motivated by the DARPA-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 two months prior to the newswire date, and terrorist events involving military targets and personnel were not considered relevant.

The MUC-4 systems had to generate one or more "templates" for each text. A *template* is essentially a large structure with a predefined set of slots, one 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. For each relevant terrorist incident in a text, the system had to instantiate a template with information about the incident. Each instantiated template contained information pertaining to a single terrorist incident. If a text described multiple relevant events, then the system was supposed to generate multiple templates. If a text described no relevant terrorist events, then the system was supposed to generate a dummy template containing no information.

DARPA provided the MUC-4 participants with a corpus of 1500 texts and associated answer keys to use for development purposes. The answer keys were instantiated templates that had been manually encoded by the participants of MUC-3[1] and MUC-4. Each answer key contained the correct information corresponding to a relevant terrorist incident reported in a text, that is, the information that *should* be extracted. DARPA 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 article.

For the purpose of text classification, the answer keys serve only 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[2] because they contained words associated with terrorism. However, many of them did not mention any relevant terrorist incidents. Of the 1700 texts in the MUC-4 corpus, only 53% described a relevant terrorist event.[3]

Because many of the texts in the corpus were irrelevant, the MUC-4 systems had to distinguish the relevant from the irrelevant texts. Although the MUC-4 task was information extraction, *information detection*[4] (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. Our MUC-4 system did not use a separate text classification module. Instead, we extracted information from every text and relied on a discourse analysis module to discard irrelevant templates. This strategy was very effective,[5] but it was expensive. A reliable text classification module could have filtered out irrele-

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

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

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

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

[5] The MUC-4 systems were evaluated on the basis of two blind test sets, TST3 and TST4. Our 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].

vant texts, so we would not have needed to apply our complete NLP system to every text.[6] 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, our discourse analysis module was domain specific and not portable across domains. Our MUC-4 experience convinced us that text classification is useful not only for stand-alone applications, but also as a partner for other natural language processing tasks.

3. THREE TEXT CLASSIFICATION ALGORITHMS BASED ON INFORMATION EXTRACTION

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

(1) A human reader will inevitably find some texts difficult to classify because those texts 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.[7] 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) Often, a single relevant sentence is 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.[8]

---

[6] The text classification algorithms described in this article would still require CIRCUS to extract information from the texts, but our complete MUC-4 system contained additional components for discourse analysis that would not need to be invoked.

[7] In an informal experiment, we asked two graduate students to scan 100 MUC-4 texts and pick out any texts that they could identify *quickly* and confidently 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 results support our claim that, in this domain at least, many documents can be accurately classified by text skimming.

[8] 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.

3.1 The Relevancy Signatures Algorithm

Although keywords are useful as a first approximation for discriminating between relevant and irrelevant texts, they do not capture the 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" refer frequently 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 in Latin American countries. In fact, every occurrence of "was found dead" in the MUC-4 corpus appears in a relevant text. Therefore, although the word "dead" is not a good keyword by itself, it *is* useful for recognizing relevant texts when it appears in certain expressions.

We see a similar phenomenon 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 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 recognize phrases approximately (e.g., Dillon [1983] and 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, and prepositional phrases. Fagan [1989] reported that syntactic analysis would have generated better phrases for indexing and eliminated many of the false hits produced by his system.

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. We represent linguistic expressions as "signatures," use statistical techniques to identify signatures that are highly correlated with relevant documents, and then use these signatures as indices to classify new texts.

3.1.1 *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." Using this representation, we can distinguish among 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 we generate a good set of relevancy signatures using a training corpus and how we use them to classify new texts.

3.1.2 *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 3 shows the steps involved in the training phase.

We hand off each text in the training set to CIRCUS, which generates a set of instantiated concept nodes. For each instantiated concept node, we create a signature by pairing the concept node with the word that triggered it. Then we compile statistics for each signature to determine how often it appeared in a relevant text. More specifically, for each signature we estimate the conditional probability that a text is *relevant*, given that it contains the signature. The formula is:

$$\Pr\left(\frac{\textit{text is relevant}}{\textit{text contains } sig_i}\right) = \frac{N_{sig_i \in REL\text{-}TEXTS}}{N_{sig_i}}$$

where N_{sig_i} is the number of occurrences of the signature sig_i in the training set, and $N_{sig_i \in REL\text{-}TEXTS}$ 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 I shows 12 signatures, their estimated conditional probabilities based on a training set of 1500 texts, and examples of sentences that will generate the signatures.

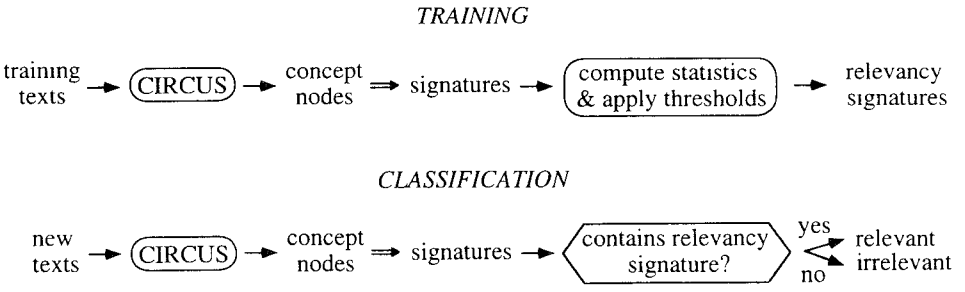


Fig. 3. Flowchart for the Relevancy Signatures Algorithm.

Table I. Sample Signatures and Conditional Probabilities

Signature	Prob.	Examples
<assassination, $murder$>	.84	the assassination of Hector Oqueli
<assassinations, $murder$>	.49	there were 2,978 political 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

The probabilities in Table I are not always intuitive. For example, 84% of texts containing the word "assassination" were relevant, but only 49% of 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 our domain of terrorist event descriptions, a text is only relevant if it describes a *specific* terrorist incident. The singular form, "assassination," is often used to report a specific assassination of a person or group of people. But the plural form, "assassinations," is often used in reference to assassinations *in general*, e.g., "The FMLN has claimed responsibility for many kidnappings and assassinations."

Table I 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 our approach is that these distinctions are identified automatically using statistics generated from a training corpus.

To select a final set of relevancy signatures, we use two thresholds: R and M. We define a "relevancy signature" as a signature that appears at least M times in the training corpus with conditional probability greater than or equal to 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, $R = 0.85$ 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 that we do not consider a signature to be "reliable" unless we have 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 the 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 then presumably very dependable, so the resulting relevant classifications are likely to be accurate. On the other hand, if we decrease R and M, then we loosen the criteria for reliability, and more signatures will be labeled as relevancy signatures. Although more texts will be classified as relevant, we also expect to see more false hits.

The second step of the algorithm is the classification phase shown in Figure 3. Given a new text to classify, first we process the text using CIRCUS, which generates a set of instantiated concept nodes. Then we create a list of signatures by pairing the concept nodes with their trigger words. If any of the signatures is a *relevancy signature*, then we classify the text as relevant. Otherwise, we classify the text 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.

3.1.3 *Experimental Results.* To evaluate the performance of the relevancy signatures algorithm, we used the 1500 texts from the MUC-4 development corpus for training[9] 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 our NLP system and our text classification algorithms. First, we processed each text in the training set using CIRCUS and compiled statistics for each signature. Next, we tested the algorithm on the two sets, TST3 and TST4. Since the relevancy signatures algorithm depends on two thresholds, R and M, we tried a variety of threshold settings. The range of threshold values was based on our experience with the algorithm and the corpus, but the values were admittedly arbitrary. We varied R from 0.70 to 0.95 in

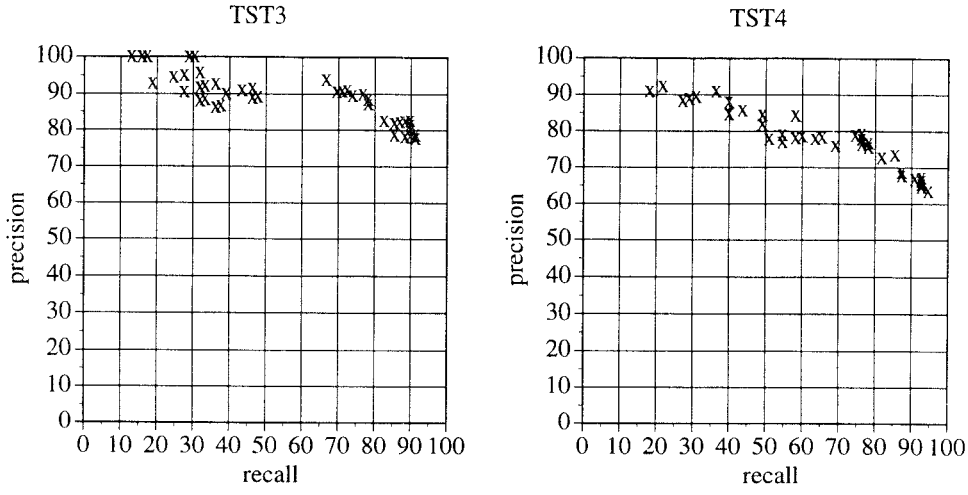
[9]These were the DEV, TST1, and TST2 texts.

increments of 0.05, and we varied M from 0 to 20 in increments of 1. Therefore, we ran the algorithm 126 times on each test set. In Section 4, we describe a more comprehensive set of experiments that addresses specifically the problem of how to find good threshold values empirically.

Figure 4 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 are collapsed into a single point on the graph. We evaluated the algorithm on the basis of two standard information retrieval metrics: recall and precision. *Recall* measures the percentage of relevant texts that are correctly classified as relevant by the algorithm. *Precision* measures the percentage of texts classified as relevant that actually *are* relevant.

In general, the data points toward the right side of the graphs correspond to threshold settings with lower values of R and M. The most obvious pattern in these graphs is the recall/precision tradeoff. As we increase R and M, 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). 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, but 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. Therefore, 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 graphs, the data points correspond to higher threshold values and hence higher levels of precision. We achieve 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 also 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, a standard information retrieval metric that combines recall and precision into a single number, was used in the final MUC-4 evaluation [MUC-4 Proceedings 1992]. The F-measure accepts a β-value that adjusts the relative weighting (importance) of recall and precision. For example, $\beta = 1.0$ gives recall and precision equal weighting, $\beta = 0.5$ makes recall half as important as precision, and $\beta = 2.0$ makes recall twice as important



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

Fig. 4. Relevancy signatures results on TST3 and TST4.

as precision. 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. 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. The beta values cover a spectrum from highly weighted recall ($\beta = 2.0$) at one end to strongly weighted precision ($\beta = 0.2$) at the other end. Since our algorithms focus on high precision, we are mostly interested in the latter end of the spectrum. Figure 4 shows that relevancy signatures achieve good performance on TST3 across the board. We achieve 100% precision on TST3 with 30% recall and still achieve 94% precision with 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. However, because the thresholds may have to be set fairly high to achieve good precision across different tests sets, consistent high precision may be possible only at relatively low recall levels.

3.1.4 *A Simple Word-Based Algorithm.* Relevancy signatures perform quite well on TST3 and TST4, but we were curious to see whether 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 for signatures. For each word[10] that appears in the training set, we count how many times it appears in the training set and how often it appears in a *relevant* text in the training set. Then for each word we estimate the conditional probability that a text is relevant given that it contains the word. The formula is

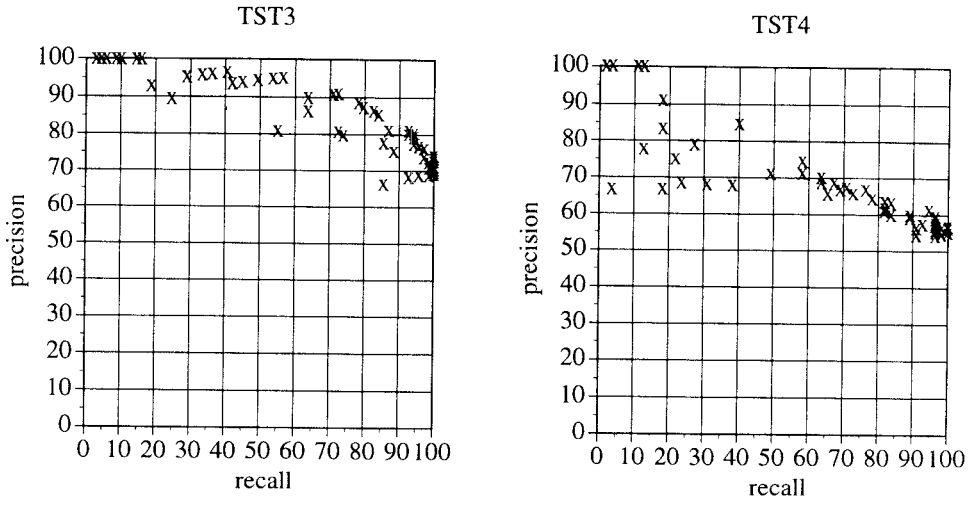
$$\Pr\left(\frac{\textit{text is relevant}}{\textit{text contains word}_i}\right) = \frac{N_{word_i \in REL\text{-}TEXTS}}{N_{word_i}}$$

where N_{word_i} is the number of times that $word_i$ appeared in the training set, and $N_{word_i \in REL\text{-}TEXTS}$ is the number of times that $word_i$ appeared in relevant texts in the training set. We then use two thresholds, R and M, to select a set of words that are highly correlated with relevance. We consider a word to be a *relevant word* if (1) its conditional probability is greater than or equal to R and (2) it appeared at least M times in the training set. Finally, we classify a new text as relevant if it contains any of the *relevant words*.

We tested this algorithm using the same training and test sets as before. We varied R from 0.70 to 0.95 in increments of 0.05, and we varied M from 0 to 50 in increments of 5. Figure 5 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 90% or higher precision for recall levels under 50%. However, precision falls quickly at the high-recall end, down to 69% at 100% recall (where there are multiple data points) 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 simple word-based algorithm has much more difficulty. Precision levels are low across the board except at the extreme low-recall end, where there are data points 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 classified correctly 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 consist-

[10] To give this algorithm the same advantages that the relevancy signatures algorithm had, we ran each text through our 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.



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

Fig. 5. Simple keyword algorithm on TST3 and TST4.

ently obtain high precision. The difference between our word-based approach and the information extraction approaches will become even more pronounced in the next two sections.

3.2 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.

In short, 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 while (b) does not. The identity of the perpetrator (rebels vs. inmates) is critical in distinguishing a terrorist event from a nonterrorist 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.

3.2.1 *Augmented Relevancy Signatures.* A relevancy signature represents the presence of a key phrase in a text. By looking only for the *existence* of a concept node, we are ignoring 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.*

We represent each slot filler as a triple of the form: (concept node type, slot name, semantic feature). For example, suppose a kidnapping concept node extracts the victim "the mayor of Achi." The victim slot filler yields the following *slot triple*: (kidnapping, victim, GOVERNMENT-OFFICIAL), because the word "mayor" is tagged with the semantic feature GOVERNMENT OFFICIAL in the dictionary.[11] 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 in the text.

An *augmented relevancy signature* is the combination of a signature *and* a slot triple that are independently highly correlated with relevance for a domain. If a text contains an augmented relevancy signature, then it must contain 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, since both sources of information must be highly correlated with relevance independently. As a result, augmented relevancy signatures are very effective at classifying texts with high precision.

3.2.2 *The Algorithm.* The augmented relevancy signatures algorithm is the same as the relevancy signatures algorithm except that we collect and apply statistics for slot triples as well as for signatures. For each concept node produced by CIRCUS, we generate a signature and a set of slot triples to represent the slot fillers extracted by the concept node. For each slot triple,

[11] 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.

we estimate the conditional probability that a text is relevant given that the slot triple appears in the text. Finally, we identify a set of "reliable" slot triples 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 greater than or equal to R_{slot}. Figure 6 illustrates the training procedure.

To classify a new text, we process the text using CIRCUS and generate a signature and set of slot triples for each concept node produced by CIRCUS. If any concept node yields both a relevancy signature *and* a reliable slot triple, then we classify the text as relevant. Otherwise we classify the text as irrelevant.

3.2.3 *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 each of R and R_{slot} from 0.70 and 0.95 in increments of 0.05, and each of M and M_{slot} from 0 to 20 in increments of 5.[12]

Figure 7 shows the scatterplots for the augmented relevancy signatures algorithm on TST3 and TST4. Each data point represents the application of the algorithm using a specific 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 threshold settings 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 3.1.4, 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.

We see several important 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 with only 30% recall. The augmented relevancy signatures algorithm also achieves 100% precision with 27% recall on TST4, while relevancy signatures alone could achieve only 93% precision with 24% recall on TST4.

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 for this is subtle. The

[12] 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.

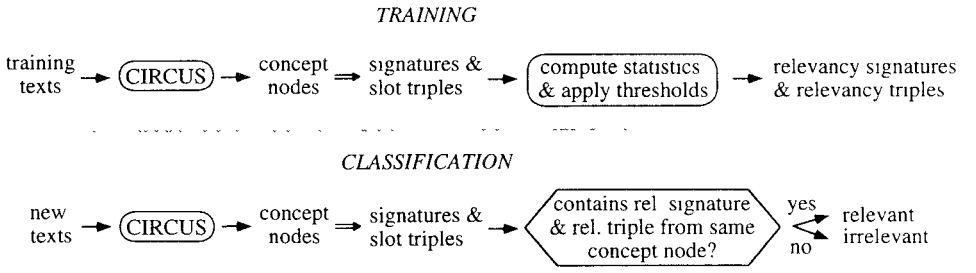
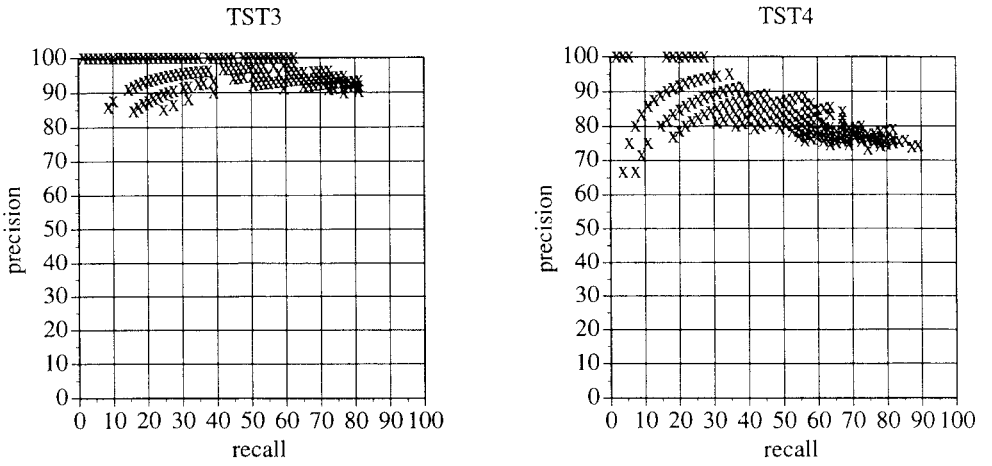


Fig 6. Flowchart for the augmented relevancy signatures algorithm.



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

Fig 7. Augmented relevancy signatures results on TST3 and TST4.

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 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 very 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.

3.3 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 keyword or phrase. Augmented relevancy signatures were our 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 describe a relevant terrorist incident even though they do not contain any specific words or phrases that are highly correlated with relevance. Some sentences contain multiple pieces of information that are highly correlated with relevance *together* but are not necessarily highly correlated with relevance 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, we would not want to classify every text that mentions a government official as a relevant text. To classify texts reliably, 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 in nature.

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 we cannot always rely on it 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, almost any reference to a victim will do. For example, if a text describes the assassination of an individual, it is probably describing a specific terrorist attack. Once again, the key phrase and the slot filler are not highly correlated with relevance independently, so we need *both* pieces of information 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.

In summary, the combination of a key phrase and a slot filler may be highly correlated with relevance *together* even though they are not necessarily correlated with relevance independently. 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 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 describes a specific terrorist incident. However, none of the words or phrases are necessarily relevant on their own. The words "killed," "wounded," and "attack" describe a violent incident that could easily be military. 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. We need to know that there was 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 classify the text as relevant with confidence.

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] and Kolodner and Simpson [1989]). By representing natural language contexts as cases, information that spans multiple clauses can be used collectively to classify texts.

3.3.1 *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,[13] 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 pertaining to the fillers extracted by these concept nodes.[14] The concept nodes extract specific strings from the text (e.g.,

[13] 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].

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

"the mayor"), but only the semantic features associated with the strings are stored in the case (e.g., GOVERNMENT-OFFICIAL). Figure 8 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 slot fillers. For example, the case in Figure 8 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 our algorithm could search for relationships between any signature and filler.

3.3.2 *The Case Base.* During training, we convert each document in the training corpus into a set of cases and then store them in a case base. Each document is represented as a set of cases, one for each sentence that produced at least one concept node. 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 document, we convert the document into cases and determine whether any of its cases are relevant to the domain. The heart of the algorithm is its ability to judge the relevancy of new cases accurately. If any of the cases are deemed to be relevant then we classify the document as relevant; otherwise we classify the document as irrelevant.

To determine the relevancy of a new case, the most obvious approach is 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 very similar cases and apply them directly to the current case (e.g., Ashley [1990] and 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, 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.[15] We do not know which cases contain the information that is responsible for the relevant classification of the document.

To get around this problem, we rely 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. We probe the case base with a set of features, retrieve cases that share these features, and

[15] 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.

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)

Fig. 8. A sample sentence, concept nodes, and resulting case

look at the statistical properties of the retrieved cases. If a high percentage of the retrieved cases come from relevant documents, we assume that this is not a coincidence. The retrieved cases must share something that makes them relevant. Since we know that all the cases 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 from the case base all cases that share exactly the same features as the new case. However, our case representation is rich enough and the training corpus is not large enough for the case base to contain many exact matches. Since we rely on the statistical properties of the retrieved cases to determine whether the cases are truly relevant, we need to retrieve a reasonably large number of cases. So instead of looking for exact matches, we use *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.

3.3.3 *Relevancy Indices.* A *relevancy index* is a collection of features that, together, reliably predict a relevant event description. To make things less abstract, first we will 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).[16] The third part of a relevancy index is the *case outline*. An outline is a list of slots in a case 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 we do not include it as part of the outline. The case outline represents the *types* of information that were found in a sentence.

By combining a signature, slot filler, and case outline into a single index, we retrieve cases that share similar key phrases and at least one piece of similar information, and that contain roughly the same amount and types of information. By indexing simultaneously on a signature and a slot filler, we retrieve cases that are strongly associated with terrorism because the *combination* of a key phrase and a 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 both the amount of information in a sentence and the types of information. Intuitively, we include the case outline as part of a relevancy index 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 II.

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 found that only 68% of the retrieved cases came from relevant texts. The only difference between these indices was the signature. Once again, this is not surprising, since the word "killed" is not a good keyword for terrorism in our corpus. In particular, the MUC-4 corpus contains many texts that contain summary descriptions of terrorist activity over a period of time, such as: *many people have been killed since 1980.* However, we can distinguish these summary descriptions from specific event descriptions by merely 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

[16] 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 slot representation for relevancy indices because they already include a signature (which represents a concept node and therefore an event type).

Table II. The Power of the Case Outline

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

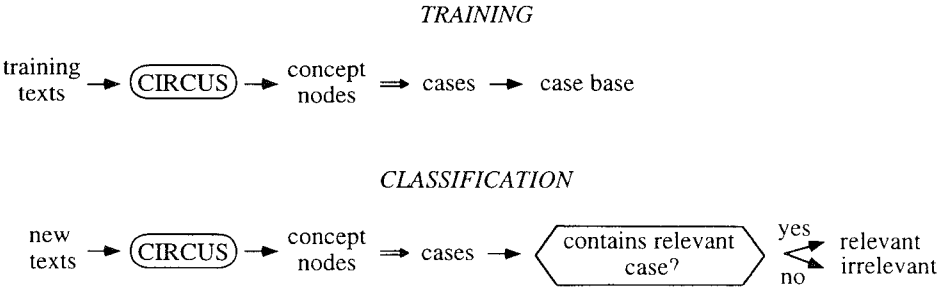


Fig. 9. Flowchart for the case-based text classification algorithm.

must contain victims *and* perpetrators. General summary descriptions usually do not mention a perpetrator, but specific event descriptions typically mention a perpetrator in conjunction with the attack. 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.

3.3.4 *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 9.

To classify a new document, we convert the concept nodes produced by CIRCUS into a set of cases. If any of the cases are judged to be relevant, then we classify the text as relevant; otherwise we classify it as irrelevant. We classify a *case* as relevant if and only 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, we first generate *all possible* relevancy indices for the case. Most cases have multiple signatures and slot fillers, so there are many possible relevancy indices for a case.[17] For each relevancy index, we retrieve all cases from the case base that share the same index. Then we estimate the conditional probability that a case is relevant given that it shares this index. The formula is

$$\Pr\left(\frac{\textit{case is relevant}}{\textit{case matches } r_i}\right) = \frac{N_{C_{r_i} \in REL\text{-}CASES}}{N_{C_{r_i}}}$$

where $N_{C_{r_i}}$ is the number of retrieved cases, and $N_{C_{r_i} \in REL\text{-}CASES}$ 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.

We use two thresholds to determine whether the probability is "high" enough: a relevancy threshold, R_{cases}, and a frequency threshold, M_{cases}. if the conditional probability is greater than or equal to 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 in a case 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

[17] 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.

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.

This sentence yields the following case:

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

The case representations 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." Our 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 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.

We use two additional "irrelevance" thresholds, I_{sig} and I_{slot}, to identify "bad" signatures and slot fillers. Given a case, we check each signature in the case by retrieving all cases from the case base that contains the signature. Then we estimate the conditional probability that a case is relevant given that it contains the signature, i.e.,

$$\Pr\left(\frac{case\ is\ relevant}{case\ contains\ sig_i}\right) = \frac{N_{C_{sig_i} \in REL\text{-}CASES}}{N_{C_{sig_i}}}$$

where $N_{C_{sig_i}}$ is the total number of retrieved cases that contain sig_i, and $N_{C_{sig_i} \in REL\text{-}CASES}$ is the number of retrieved cases that contain sig_i that come from relevant texts. If the probability is less than 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, we check each slot filler in the case by retrieving from the case base all cases that also contain the slot filler and the same case outline. Then we estimate the conditional probability that a case is relevant given that it contains the slot filler and outline; i.e.,

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

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

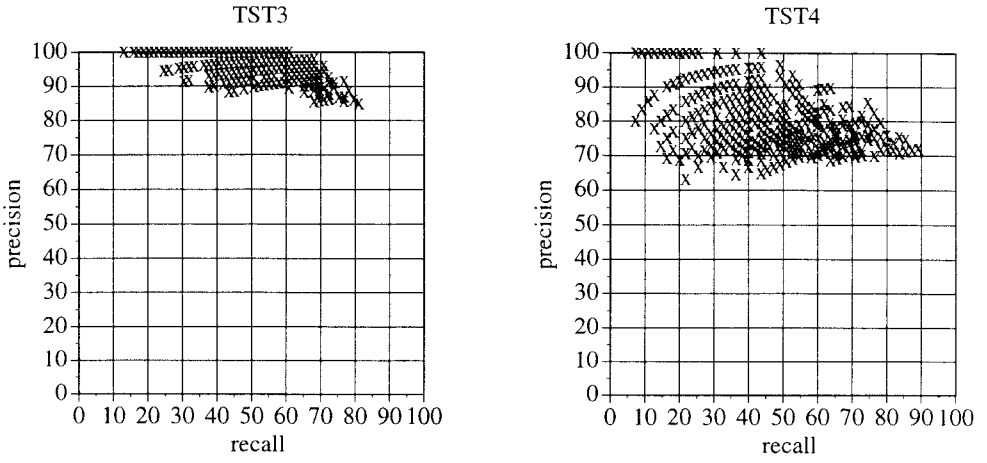
These four thresholds allow the user to tailor the performance of the algorithm for his or her domain. By increasing I_{sig} and I_{slot} the user can adjust the sensitivity of the algorithm to irrelevance cues. The user can also adjust M_{cases} based on the size of the training corpus. For a large 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 will contain relatively few cases. We will address the issue of selecting appropriate threshold values more thoroughly in Section 4.

3.3.5 *Experimental Results.* We evaluated the case-based 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 0.70 to 0.95 in increments of 0.05, M_{cases} from 0 to 20 in increments of 2, and gave each of I_{sig} and I_{slot} the values $\{0.50, 0.60, 0.70\}$.

Figure 10 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, while the augmented relevancy signatures algorithm reached only 27% recall with 100% precision. These results support our claim that using natural language contexts allows us to classify texts correctly that are inaccessible to more simple word- and phrase-based techniques. 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(0.5) column (68% recall with 98% precision).

Note that we see more data points with relatively low precision on TST4. This is because combining low threshold values with larger natural language contexts for classification can work against the user. 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, we need to choose appropriate threshold settings. In the next section, we describe a procedure that allows the user to find good threshold values empirically.



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

Fig. 10. Case-based text classification results.

4. COMPARATIVE ANALYSIS

In the previous section, we presented three text classification algorithms and showed how performance improves as we use additional natural language context to classify texts. However, all of 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 positively or negatively, and the texts may not be representative of the corpus in general. Unfortunately, the MUC-4 corpus is relatively small, and since our algorithms all depend on statistical data, we must reserve a large number of texts for training.

To evaluate the algorithms on the basis of more texts, we used a 10-fold cross-validation design. This experimental design uses *all* of the texts in the corpus for testing by rotating different subsets of the corpus for training. First, we divide randomly 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 11 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

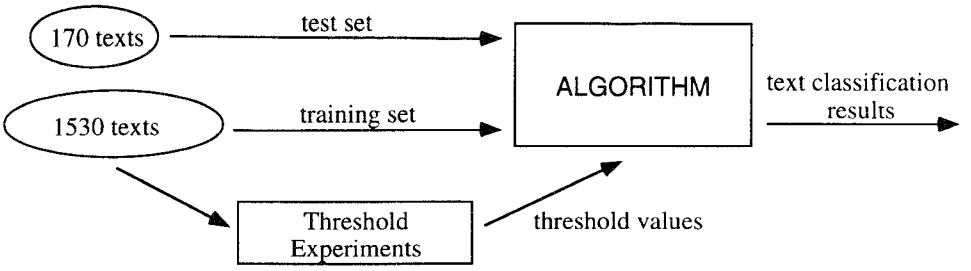


Fig. 11. One fold of cross-validation.

test set. Finally, we combine (sum) the results over all 10 folds. For example, in a 2-fold cross-validation design, if fold #1 classified 24 of 50 texts correctly and fold #2 classified 16 of 50 texts correctly 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).

An important issue that still needs to be addressed is how to find good 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 lower threshold values, and a user who demands high precision should choose higher threshold values.

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 setting empirically. The procedure recommends threshold settings that are appropriate for a variety of metrics. By using multiple metrics, we create a sort of recall/precision "knob." The user can look at the results obtained on the training set under each metric and choose threshold settings that will achieve 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 are threshold values that achieved the desired behavior on training texts.

We used seven metrics to represent a spectrum that shifts importance gradually from recall to precision. At the two ends of the spectrum, we used metrics to 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 used the F-measure with different β-values to vary the relative importance of recall and precision gradually. As before, we chose five values of β: 2.0, 1.0, 0.5, 0.3, 0.2. Note that several β-values give extra weighting to precision because our algorithms were specifically designed for high-precision text classification.

Figure 12 shows the procedure for automatically deriving good threshold values. This design mimics the 10-fold cross-validation experiment. Using the

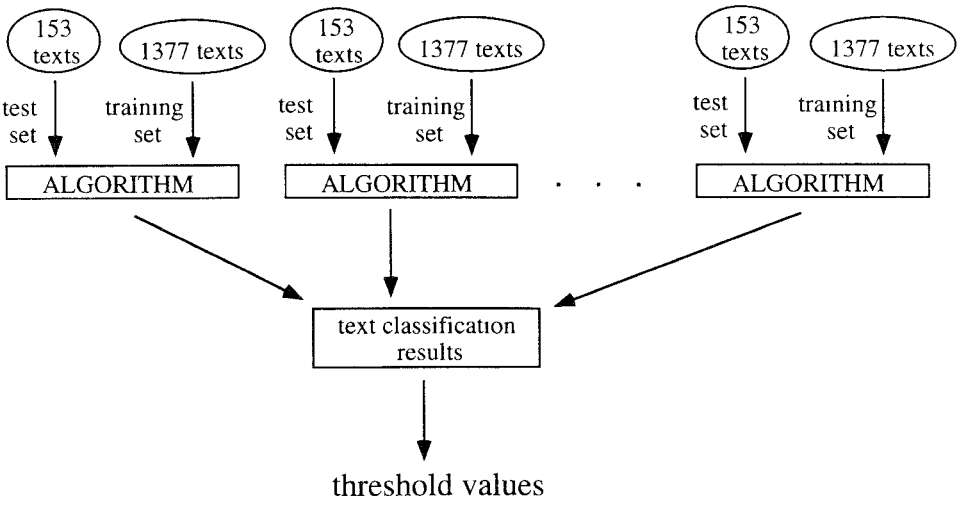


Fig. 12. Threshold experiments

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 setting.[18] Then we sum the results of all the folds for each set of threshold values. For example, we would sum all of the classification results for the relevancy signatures algorithm with threshold values: {0.85, 10}. This sum reflects the performance of the algorithm on all 1530 texts using threshold values {0.85, 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 11 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 derive a good

[18] 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.

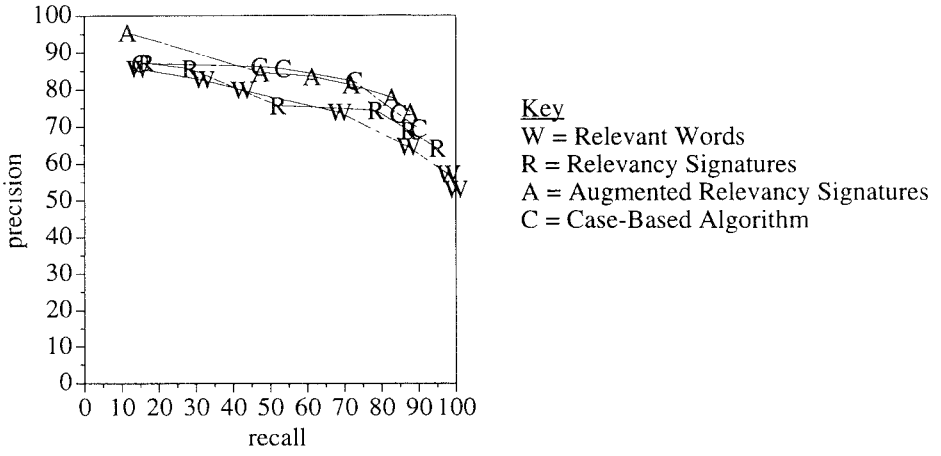
set of threshold values empirically. If we assume that the training texts are representative of new texts, the results obtained in the threshold experiments predict the level of performance that will be achieved on novel texts.

Finally, we evaluate the test sets for each fold of the cross-validation experiment 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 of 1700 texts. These data points are shown in Figure 13. In addition to the three text classification algorithms described in Section 3, we also evaluated the *relevant words* algorithm described in Section 3.1.4. This algorithm serves as a sort of baseline for the performance of a simple word-based approach on this corpus. Note that some of the algorithms generated fewer than seven data points because multiple metrics produced the same results.

It is reassuring to see that the results in Figure 13 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 of all of the algorithms, which is not surprising since they have 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 from 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 outperforming another is compelling, as is the consistency of these results with those reported in the earlier experiments.

Finally, the performance levels shown in Figure 13 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



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

Fig. 13. Graph of cross-validation results

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.

5. PORTING THE TEXT CLASSIFICATION SYSTEM TO NEW DOMAINS

In Section 1, we claimed that our text classification system is portable across domains. The classification algorithms, themselves, use general statistical techniques that are not domain dependent in any way. The underlying information extraction system, however, uses two domain-specific dictionaries, and the text classification algorithms depend on a domain-specific training corpus. In this section, we describe briefly how to acquire the domain-specific dictionaries with a minimal amount of effort and how to generate an appropriate training corpus.

5.1 Porting the Information Extraction System

The concept node dictionary is potentially the main knowledge-engineering bottleneck for the information extraction system. For MUC-4, we defined manually a dictionary of concept nodes for the domain of terrorism. We estimate that it took approximately 1500 person-hours on the part of highly trained researchers to build the dictionary. However, we have shown else-

where [Riloff 1993a; Riloff and Lehnert 1993a] that concept node definitions can be acquired automatically given an appropriate training corpus. Using a system called AutoSlog, in only 5 person-hours we automatically generated a concept node dictionary for terrorism that achieved 98% of the performance of the hand-crafted dictionary.[19] Furthermore, AutoSlog can be used by anyone who is familiar with the domain after only a minimal amount of training [Riloff and Lehnert 1993b].

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 name of each perpetrator, victim, target, and weapon involved in a relevant terrorist event should be tagged with its 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 AutosSlog, only the targeted information needs to be tagged. Sentences, paragraphs, and even texts that are irrelevant to the domain can be effectively ignored.

CIRCUS also relies on a lexical dictionary for part-of-speech tags and semantic features. Machine-readable dictionaries are available that contain part-of-speech tags for thousands of words. An alternative approach is to use a part-of-speech tagger that relies on statistics generated from a corpus. Several statistically based part-of-speech taggers have been developed, including OTB [Lehnert et al. 1993] and POST [Weischedel et al. 1993].

The relevancy signatures algorithm does not rely on semantic features at all,[20] but the augmented relevancy signatures algorithm and the case-based algorithm do. Some techniques have been developed to acquire semantic features dynamically during text processing (e.g., 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.[21]

5.2 Generating a Training Corpus

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

[19] We should note, however, that AutoSlog was not designed to generate concept nodes that represent irrelevancy cues, such as the ones described in Section 3.3.4.

[20] 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, concept node definitions produced by AutoSlog *do not* use semantic features.

[21] We used this strategy to acquire semantic features for the business domain of joint ventures, and it took us only a few hours to generate a core dictionary.

both relevant and irrelevant texts. If the corpus contains mostly relevant tests, 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 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 refer frequently 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 prefiltered 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 corpus. The natural language processing techniques can be applied to this prefiltered corpus to generate more accurate classifications.

5.3 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 was performed on a Decstation 5000/240 running Ultrix 4.2 with 64 MB of RAM.

6. CONCLUSIONS

We have presented three algorithms that use information extraction to achieve high-precision text classification. In general, we obtained better performance as we used more extracted information. However, the algorithms

have different properties that make each of them suitable for different types of domains. The relevancy signatures algorithm uses the least amount of extracted information. This algorithm is suitable for domains that are characterized by strong key phrases and in which surrounding context is usually not important. The augmented relevancy signatures algorithm is more appropriate for domains with strong key phrases when surrounding context *is* an important factor. This algorithm emphasizes high precision more strongly than the others. Finally, the case-based algorithm should be used for domains that may not have strong key phrases and for which context is crucial.

Our work on text classification differs from previous work in several respects. Within the IR community, text classification has typically been approached using word-based techniques and statistical methods (e.g., Maron [1961], Borko and Bernick [1963], and Hoyle [1973]). As we explained in Section 1, word-based techniques have many limitations. Knowledge-based text classification systems have been developed that address some of these problems using rule bases or other knowledge sources (e.g., Goodman [1991], Hayes and Weinstein [1991], and Rau and Jacobs [1991]). For example, CONSTRUE [Hayes and Weinstein 1991] uses a rule base to identify phrases, context, and even text structure. The rules can approximately recognize language constructs and contexts without requiring the use of a natural language processing system. The main advantage of our approach over other knowledge-based systems is that our system can be easily ported to new domains. Most knowledge-based systems require an extensive manual knowledge-engineering effort that can take significant time and human resources. Additionally, our system carries with it the benefits of natural language processing. There have been other systems that use NLP techniques as a basis for information retrieval tasks (e.g., Lewis et al. [1989], Mauldin [1991], and Rau and Jacobs [1991]). For example, FERRET [Mauldin 1991] used information extraction techniques to recognize relevant documents in a standard information retrieval task. However, these NLP systems also require extensive manual knowledge engineering to build domain-specific dictionaries and knowledge structures.

Another important distinction is that our algorithms do not use the entire representation of a text to decide whether the text is relevant. Although we skim the entire text to extract relevant information, in general we do not use all of this information to make a classification. Instead, we look in the text for "hot spots" that are highly correlated with relevance. Once we find a highly relevant phrase or context, we immediately classify the text as relevant. This is fundamentally different from other systems that reason with the entire representation of a text. For example, Maron [1961] also classified texts by estimating the conditional probability of a category. But he estimated the conditional probability of a category given *all* of the indexing terms found in the document. Our algorithms estimate the conditional probability of a category given an *individual* indexing term. Furthermore, our 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 more complex indexing terms, we can use simple statistical techniques to identify reliable phrases and contexts that can be used effectively to achieve high precision.

In summary, we have proposed 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 performances than either technology could support on its own.

ACKNOWLEDGMENTS

The authors would like to thank Jamie Callan, Claire Cardie, and Jon Peterson for providing helpful comments on earlier drafts of this article.

REFERENCES

ASHLEY, K 1990 *Modelling Legal Argument: Reasoning with Cases and Hypotheticals*. The MIT Press, Cambridge, Mass.

BELKIN, N., AND CROFT, W. B. 1992. Information filtering and information retrieval: Two sides of the same coin? *Commun. ACM 35*, 12 (Dec.), 29–38.

BORKO, H., AND BERNICK, M. 1963 Automatic document classification. *J. ACM 10* 2, 151–162.

CARDIE, C. 1993. A case-based approach to knowledge acquisition for domain-specific sentence analysis. In *Proceedings of the 11th National Conference on Artificial Intelligence*. AAAI Press, Menlo Park, Calif., 798–803.

CROFT, W. B., TURTLE, H. R. AND LEWIS, D. D. 1991. The use of phrases and structured queries in information retrieval. In *Proceedings of SIGIR 1991*. ACM, New York, 32–45.

DILLON, M. 1983 FASIT: A fully automatic syntactically based indexing system. *J. Am Soc Inf. Sci. 34*, 2, 99–108.

FAGAN, J. 1989. The effectiveness of a nonsyntactic approach to automatic phrase indexing for document retrieval *J. Am. Soc. Inf. Sci. 40*, 2, 115–132.

FRANCIS, W., AND KUCERA, H. 1982. *Frequency Analysis of English Usage* Houghton Mifflin, Boston, Mass.

GOODMAN, M. 1991. Prism: A case-based telex classifier. In *Proceedings of the 2nd Annual Conference on Innovative Applications of Artificial Intelligence*. AAAI Press, Menlo Park, Calif., 25–37.

HAMMOND, K. 1986. CHEF: A model of case-based planning. In *Proceedings of the 5th National Conference on Artificial Intelligence*. Morgan Kaufmann, San Mateo, Calif., 267–271.

HARMAN, D., ED. 1993. *The First Text Retrieval Conference (TREC1)*. Spec. Pub 200–207, National Inst. of Standards and Technology, Gaithersburg, Md

HAYES, P. J., AND WEINSTEIN, S. P. 1991. Construe-TIS: A system for content-based indexing of a database of news stories. In *Proceedings of the 2nd Annual Conference on Innovative Applications of Artificial Intelligence*. Menlo Park, Calif. AAAI Press, 49–64.

HOYLE, W 1973. Automatic indexing and generation of classification systems by algorithm *Inf. Storage Retriev. 9*, 4, 233–242.

IWANSKA, L., APPELT, D , AYUSO, D., DAHLGREN, K., GLOVER STALLS, B., GRISHMAN, R., KRUPKA, G., MONTGOMERY, C., AND RILOFF, E. 1991. Computational aspects of discourse in the context of MUC-3. In *Proceedings of the 3rd Message Understanding Conference (MUC-3)*. Morgan Kaufmann, San Mateo, Calif., 256–282.

KOLODNER, J., AND SIMPSON, R. 1989. The MEDIATOR: Analysis of an early case-based problem solver. *Cog. Sci. 13*, 4, 507–549.

KROVETZ, R. AND CROFT, W. B. 1989. Word sense disambiguation using machine-readable dictionaries. In *Proceedings of SIGIR 1989*. ACM, New York.

LEHNERT, W. 1991. Symbolic/subsymbolic sentence analysis: Exploiting the best of two worlds. In *Advances in Connectionist and Neural Computational Theory*, J. Barnden and J. Pollack, Eds., Vol. 1 Ablex Publishers, Norwood, N.J., 135–164.

LEHNERT, W. G. AND SUNDHEIM, B. 1991. A performance evaluation of text analysis technologies. *AI Mag. 12*, 3, 81–94.

LEHNERT, W., CARDIE, C., FISHER, D., MCCARTHY, J., RILOFF, E., AND SODERLAND, S. 1992. University of Massachusetts: Description of the CIRCUS system as used for MUC-4. In *Proceedings of the 4th Message Understanding Conference (MUC-4)*. Morgan Kaufmann, San Mateo, Calif., 282–288.

LEHNERT, W., MCCARTHY, J., SODERLAND, S., RILOFF, E., CARDIE, C., PETERSON, J., FENG, F., DOLAN, C., AND GOLDMAN, S. 1993. University of Massachusetts/Hughes: Description of the CIRCUS system as used for MUC-5. In *Proceedings of the 5th Message Understanding Conference (MUC-5)*. Morgan Kaufmann, San Mateo, Calif., 277–291.

LEWIS, D. D., CROFT, W. B., AND BHANDARU, N. 1989. Language-oriented information retrieval. *Int. J. Intell. Syst. 4*, 3, 285–318.

MARCUS, M., SANTORINI, B., AND MARCINKIEWICZ, M. 1993. Building a large annotated corpus of English: The Penn Treebank. *Comput. Ling. 19*, 2, 313–330.

MARON, M. 1961. Automatic indexing: An experimental inquiry. *J. ACM 8*, 404–417.

MAULDIN, M. 1991. Retrieval performance in FERRET: A conceptual information retrieval system. In *Proceedings of SIGIR 1991*. ACM, New York, 347–355.

RAU, L. F., AND JACOBS, P. S. 1991. Creating segmented databases from free text for text retrieval. In *Proceedings of SIGIR 1991*. ACM, New York, 337–346.

RILOFF, E. 1993a. Automatically constructing a dictionary for information extraction tasks. In *Proceedings of the 11th National Conference on Artificial Intelligence*. AAAI Press, Menlo Park, Calif., 811–816.

RILOFF, E. 1993b. Using cases to represent context for text classification. In *Proceedings of the 2nd International Conference on Information and Knowledge Management (CIKM-93)*. ACM, New York, 105–113.

RILOFF, E., AND LEHNERT, W. 1993a. Automated dictionary construction for information extraction from text. In *Proceedings of the 9th IEEE Conference on Artificial Intelligence for Applications*. IEEE Computer Society Press, Los Alamitos, Calif., 93–99.

RILOFF, E., AND LEHNERT, W. 1993b. Dictionary construction by domain experts. In *Proceedings of the TIPSTER Text Program (Phase I)*. Morgan Kaufmann, San Francisco, Calif., 257–259.

RILOFF, E., AND LEHNERT, W. 1992. Classifying texts using relevancy signatures. In *Proceedings of the 10th National Conference on Artificial Intelligence*. AAAI Press, Menlo Park, Calif. 329–334.

SALTON, G. 1989. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, Reading, Mass.

SUNDHEIM, B., Ed. 1993. *Proceedings of the 5th Message Understanding Conference (MUC-5)*. Morgan Kaufmann, San Mateo, Calif. To be published.

SUNDHEIM, B., Ed. 1992. *Proceedings of the 4th Message Understanding Conference (MUC-4)*. Morgan Kaufmann, San Mateo, Calif.

SUNDHEIM, B., Ed. 1991. *Proceedings of the 3rd Message Understanding Conference (MUC-3)*. Morgan Kaufmann, San Mateo, Calif.

TURTLE, H., AND CROFT, W. B. 1991. Efficient probabilistic inference for text retrieval. In *Proceedings of RIAO 91*. 644–661.

WEISCHEDEL, R., METEER, M., SCHWARTZ, R., RAMSHAW, L., AND PALMUCCI, J. 1993. Coping with ambiguity and unknown words through probabilistic models. *Comput. Ling. 19* 2, 359–382.