Automatically Acquiring Conceptual Patterns Without an Annotated Corpus

Ellen Riloff and Jay Shoen
Department of Computer Science
University of Utah
Salt Lake City, UT 84112
E-mail: riloff@cs.utah.edu

Abstract

Previous work on automated dictionary construction for information extraction has relied on annotated text corpora. However, annotating a corpus is time-consuming and difficult. We propose that conceptual patterns for information extraction can be acquired automatically using only a preclassified training corpus and no text annotations. We describe a system called AutoSlog-TS, which is a variation of our previous AutoSlog system, that runs exhaustively on an untagged text corpus. Text classification experiments in the MUC-4 terrorism domain show that the AutoSlog-TS dictionary performs comparably to a hand-crafted dictionary, and actually achieves higher precision on one test set. For text classification, AutoSlog-TS requires no manual effort beyond the preclassified training corpus. Additional experiments suggest how a dictionary produced by AutoSlog-TS can be filtered automatically for information extraction tasks. Some manual intervention is still required in this case, but AutoSlog-TS significantly reduces the amount of effort required to create an appropriate training corpus.

1 Introduction

In the last few years, significant progress has been made toward automatically acquiring conceptual patterns for information extraction (e.g., [Riloff, 1993; Kim and Moldovan, 1993]). However, previous approaches require an annotated training corpus or some other type of manually encoded training data. Annotated training corpora are expensive to build, both in terms of the time and the expertise required to create them. Furthermore, training corpora for information extraction are typically annotated with domain-specific tags, in contrast to general-purpose annotations such as part-of-speech tags or noun-phrase bracketing (e.g., the Brown Corpus [Francis and Kucera, 1982] and the Penn Treebank [Marcus *et al.*, 1993]). Consequently, a new training corpus must be annotated for each domain.

We have begun to explore the possibility of using an untagged corpus to automatically acquire conceptual patterns for information extraction. Our approach uses a combination of domain-independent linguistic rules and statistics. The linguistic rules are based on our previous system, AutoSlog [Riloff, 1993], which automatically constructs dictionaries for information extraction using an annotated training corpus. We have put a new spin on the original system by applying it exhaustively to an untagged but preclassified training corpus (i.e., a corpus in which the texts have been manually classified as either relevant or irrelevant). Statistics are then used to sift through the myriad of patterns that it produces. The new system, AutoSlog-TS, can generate a conceptual dictionary of extraction patterns for a domain from a preclassified text corpus.

First, we give a brief overview of information extraction and the CIRCUS sentence analyzer that we used in these experiments. Second, we describe the original AutoSlog system for automated dictionary construction and explain how AutoSlog was adapted to generate patterns from an untagged corpus. Next, we present empirical results from experiments with AutoSlog-TS using the MUC-4 text corpus.

2 Information Extraction

Information extraction (IE) is a natural language processing task that involves extracting predefined types of information from text. Information extraction systems are domain-specific because they extract facts about a specific domain and typically ignore information that is not relevant to the domain. For example, an information extraction system for the terrorism domain might extract the names of perpetrators, victims, physical targets, and weapons associated with terrorist events mentioned in a text. The information extraction task has received a lot of attention recently as a result of the message understanding conferences (MUCs) [MUC-5 Proceedings, 1993; MUC-4 Proceedings, 1992; MUC-3 Proceedings, 1991].

The systems described in this paper use a conceptual sentence analyzer called CIRCUS [Lehnert, 1991]. CIRCUS extracts information using domain-specific structures called *concept nodes.* Each concept node is triggered by a keyword, but is activated only in certain linguistic contexts. For example, a concept node called $murder-passive$ is triggered by the verb "murdered" but activated only when the verb appears in a passive construction. Therefore this concept node would be activated by phrases such as "X was murdered", "X and Y were murdered", and "X has been murdered." The subject of the verb is extracted as the victim of the murder. Figure 1 shows a sample sentence and the instantiated concept node produced by CIRCUS.

Sentence: Three peasants were murdered.

$murder-passive$
victim = "three peasants"

Figure 1: An instantiated concept node

A similar concept node called $murder-active$ recognizes active forms of the verb "murdered", such as "terrorists murdered three peasants." This concept node is also triggered by the verb "murdered", but is activated only when the verb appears in an active construction. In this case, the subject of the verb is extracted as the perpetrator of the murder.

CIRCUS relies entirely on its dictionary of concept nodes to extract information, so it is crucial to have a good concept node dictionary for a domain. However, building a concept node dictionary by hand is tedious and time-consuming. We estimate that it took approximately 1500 person-hours to construct a concept node dictionary by hand for the MUC-4 terrorism domain [Lehnert *et al.*, 1992]. Subsequently, we developed a system called AutoSlog that can build concept node dictionaries automatically using an annotated training corpus. The next section describes the original version of AutoSlog as well as the new version, AutoSlog-TS, that generates concept node dictionaries automatically using only a preclassified training corpus.

3 Automated Dictionary Construction for Information Extraction

A major knowledge-engineering bottleneck for information extraction (IE) systems is the process of constructing a dictionary of appropriate extraction patterns. A few systems have been developed recently to build dictionaries for information extraction automatically, such as AutoSlog [Riloff, 1993] and PALKA [Kim and Moldovan, 1993]. These systems generate extraction patterns automatically using a set of associated answer keys or an annotated training corpus. In this section, we describe the original AutoSlog system for automated dictionary construction and then present AutoSlog-TS, a variant of AutoSlog that does not rely on text annotations.

3.1 AutoSlog: Automated Dictionary Construction Using Text Annotations

The guiding principle behind AutoSlog is that most role relationships can be identified by local linguistic context surrounding a phrase. For example, consider the sentence "John Smith was kidnapped by three armed men." To identify "John Smith" as the victim of a kidnapping, we must recognize that he is the subject of the passive verb "kidnapped." Similarly, to identify "three armed men" as the perpetrators, we must recognize that "three armed men" is the object of the preposition "by" and attaches to the verb "kidnapped." It is impossible to look at an isolated noun phrase such as "John Smith" and determine whether he is a perpetrator or a victim without considering local context.

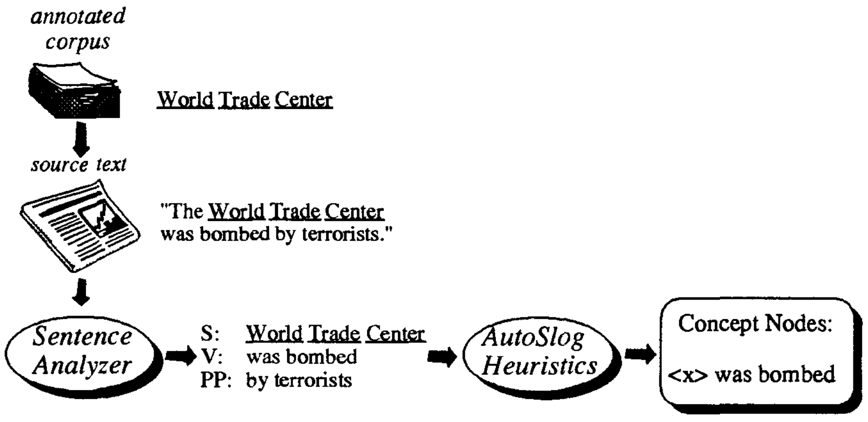

Figure 2: AutoSlog flowchart

AutoSlog uses simple domain-independent linguistic rules to create extraction patterns for a given set of noun phrases in a text corpus. Figure 2 shows the steps involved in dictionary construction. As input, AutoSlog requires a set of annotated texts in which the noun phrases that need to be extracted have been tagged.[1] For each "targeted" noun phrase, AutoSlog finds the sentence in which it was tagged[2] and passes the sentence to CIRCUS for syntactic analysis.

[1]Alternatively, a set of answer keys that list the relevant noun phrases (e.g., the MUC-4 answer keys) could be used (e.g., see [Riloff, 1993]).

[2]If AutoSlog does not know which sentence the noun phrase came from, it searches for the first sentence in the text that contains the noun phrase. We adopted this strategy in earlier experiments that used the MUC-4 answer keys as input [Riloff, 1993].

CIRCUS separates each sentence into clauses and identifies the subject, verb, direct object and prepositional phrases in each clause. AutoSlog then determines which clause contains the targeted noun phrase and whether it is a subject, direct object, or prepositional phrase.

Next, AutoSlog uses a small set of heuristics to infer which other words in the sentence identify the role of the noun phrase. If the targeted noun phrase is the subject or direct object of a clause then AutoSlog infers that the verb defines the role of the noun phrase. AutoSlog uses several rules to recognize different verb forms. In the subject case, consider the sentence "John Smith killed two people" and the targeted noun phrase "John Smith" tagged as a perpetrator. AutoSlog generates a concept node that is triggered by the verb "killed" and activated when the verb appears in an active construction; the resulting concept node recognizes the pattern "X killed" and extracts X as a perpetrator. Given the sentence "John Smith was killed" with "John Smith" tagged as a victim, AutoSlog generates a concept node that recognizes the pattern "X was killed" and extracts X as a victim.

In the direct object case, the sentence "the armed men killed John Smith" produces the pattern "killed X." If the targeted noun phrase is in a prepositional phrase, then AutoSlog uses a simple pp-attachment algorithm to attach the prepositional phrase to a previous verb or noun in the sentence which is then used as a trigger word for a concept node. For example, "the men were killed in Bogota by John Smith" produces the pattern "killed by X." It should be noted that, although we are using a simple phrase-like notation for the patterns, they are actually concept nodes activated by an NLP system so the words do not have to be strictly adjacent in the text.

Linguistic Pattern	**Example**
1. **<subject> active-verb**	<perpetrator> bombed
2. **<subject> active-verb direct-object**[3]	<perpetrator> claimed responsibility
3. **<subject> passive-verb**	<victim> was murdered
4. **<subject> verb infinitive**	<perpetrator> attempted to kill
5. **<subject> auxiliary noun**	<victim> was victim
6. **active-verb <direct-object>**	bombed <target>
7. **passive-verb <direct-object>**[4]	killed <victim>
8. **infinitive <direct-object>**	to kill <victim>
9. **verb infinitive <direct-object>**	threatened to attack <target>
10. **gerund <direct-object>**	killing <victim>
11. **noun auxiliary <direct-object>**	fatality was <victim>
12. **noun preposition <noun-phrase>**	bomb against <target>
13. **active-verb preposition <noun-phrase>**	killed with <instrument>
14. **passive-verb preposition <noun-phrase>**	was aimed at <target>
15. **infinitive preposition <noun-phrase>**[3]	to fire at <victim>

Figure 3: AutoSlog heuristics and examples from the terrorism domain

The set of heuristics used by AutoSlog is shown in Figure 3. The heuristics are divided into three categories depending upon where the targeted noun phrase is found. The location is indicated by the bracketed item (subject, direct-object, noun-phrase in a PP). The other words represent the surrounding context used to construct a concept node. The examples in the right-hand column show instantiated patterns for which AutoSlog generated concept nodes based on the

[3]These patterns were used by AutoSlog in the current experiments but not in the experiments reported in [Riloff, 1993].

[4]In principle, passive verbs should not have direct objects. However, CIRCUS occasionally confuses active and passive verb forms so we included this pattern.

general pattern on the left. The underlined word represents the trigger word, the bracketed item represents the type of information that will be extracted by the concept node, and the remaining words represent the required context.

In previous experiments, we used AutoSlog to construct a dictionary for the MUC-4 terrorism domain using 772 relevant texts from the MUC-4 corpus. AutoSlog created 1237 concept node definitions, but many of these concept nodes represented general expressions that will not reliably extract relevant information. Therefore, we introduced a human-in-the-loop to weed out the unreliable definitions. A person manually reviewed all 1237 definitions and retained 450 of them for the final dictionary. The resulting dictionary achieved 98% of the performance of a dictionary that was hand-crafted for the MUC-4 terrorism domain [Riloff, 1993].

One of the main differences between AutoSlog and previous lexical acquisition systems is that AutoSlog creates new definitions entirely from scratch. In contrast, previous language learning systems (e.g., [Jacobs and Zernik, 1988; Carbonell, 1979; Granger, 1977]) create new definitions based on the definitions of other known words in the context. That is, they assume that some definitions already exist and use those definitions to create new ones. The structures created by AutoSlog are also considerably different than the lexical definitions created by most systems, although the PALKA system [Kim and Moldovan, 1993] creates similar extraction patterns. The main difference between PALKA and AutoSlog is that PALKA is given the set of keywords associated with each concept (essentially its "trigger words") and then learns to generalize the patterns surrounding the keywords. In contrast, AutoSlog infers the trigger words and patterns on its own but does not generalize them.

3.2 AutoSlog-TS: Automated Dictionary Construction Without Text Annotations

As described in the previous section, AutoSlog requires an annotated training corpus in which the noun phrases that should be extracted have been tagged. Creating an annotated corpus is much easier than building a dictionary by hand. However, the annotation process is not trivial. It may take days or even weeks for a domain expert to annotate several hundred texts.[5] But perhaps even more importantly, the annotation process is not always well-defined; in many cases, it is not clear which portions of a text should be annotated. Complex noun phrases (e.g., conjunctions, appositives, prepositional phrases) are often confusing for annotators. Should the entire noun phrase be tagged or just the head noun? Should modifiers be included? Should prepositional phrases be included? Conjuncts and appositives? These issues are not only frustrating for a user, but can have serious consequences for the system. A noun phrase that is incorrectly annotated often produces an undesirable extraction pattern or produces no extraction pattern at all.

To bypass the need for an annotated corpus, we created a new version of AutoSlog that does not rely on text annotations. The new system, Autoslog-TS, can be run exhaustively on an untagged but preclassified corpus. None of the words or phrases in the texts need to be tagged, but each text must be classified as either *relevant* or *irrelevant* to the targeted domain.[6] Figure 4 shows the steps involved in dictionary construction. The process breaks down into two stages:

[5]In a preliminary experiment, a user annotated 160 texts in about 8 hours.

[6]It is important for the training corpus to be representative of the texts expected in the future. For text classification tasks, the irrelevant texts should reflect the types of texts that will need to be distinguished from relevant texts. For example, many of the irrelevant texts in the MUC-4 corpus describe military actions so the resulting AutoSlog-TS dictionary is especially well-suited for discriminating texts describing military incidents from those describing terrorist incidents.

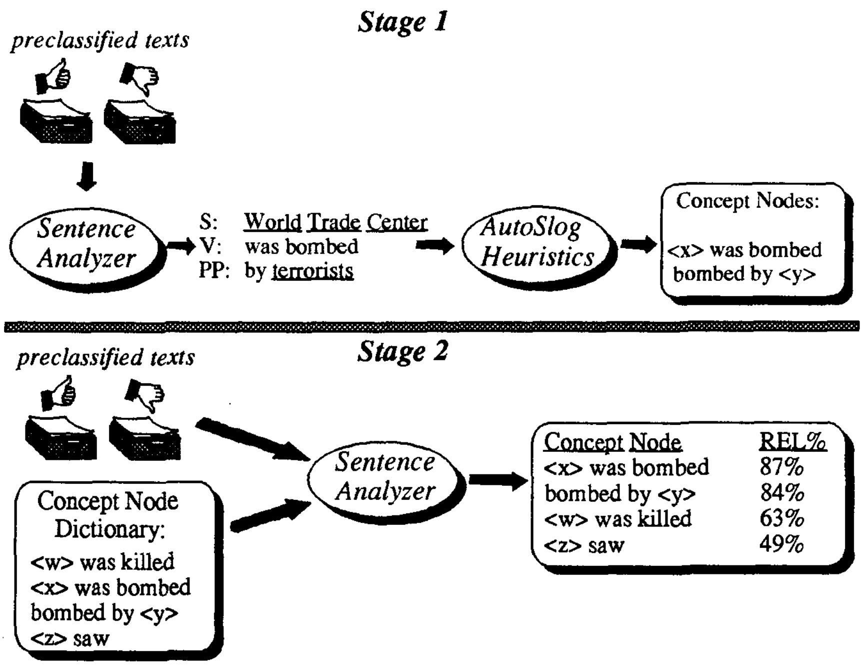

Figure 4: AutoSlog-TS flowchart

Stage 1: Generating Concept Nodes

1. Given a corpus of preclassified texts, a sentence analyzer (CIRCUS) is applied to each sentence to identify all of the noun phrases in the sentence. For example, in Figure 4, two noun phrases are identified: "The World Trade Center" and "terrorists."

2. For each noun phrase, the system determines whether the noun phrase was a subject, direct object, or prepositional phrase based on the syntactic analysis produced by the sentence analyzer.

3. All of the appropriate heuristics are fired. For example, in Figure 4 "The World Trade Center" was identified as the subject of the sentence so all of the subject patterns are fired (patterns #1-5 in Figure 3). Pattern #3 is the only one that is satisfied, so a single concept node is generated that recognizes the pattern "X was bombed." It is possible for multiple heuristics to fire; for example, patterns #1 and #2 may both fire if the targeted noun phrase is the subject of an active verb and takes a direct-object.

After processing the training texts, we have a huge collection of concept nodes. The second stage involves collecting statistics to determine which concept nodes represent domain-specific expressions.

Stage 2: Statistically Filtering the Concept Nodes

1. All of the newly generated concept nodes are loaded into the system and the training corpus is run through the sentence analyzer again. This time, however, the concept nodes are activated during sentence processing.

2. Statistics are computed to determine how often each concept node was activated in relevant texts and how often it was activated in irrelevant texts. We calculate the *relevancy rate* of each concept node (i.e., the number of occurrences in relevant texts divided by the total number of occurrences), and the frequency of each concept node (i.e., the total number of times it was activated in the corpus).

After Stage 1, we have a large set of concept node definitions that, collectively, can extract virtually[7] every noun phrase in the corpus. Most of the concept nodes represent general phrases that are likely to occur in a wide variety of texts (e.g., "X saw"). However, some of the concept nodes represent domain-specific patterns (e.g., "X was bombed"). Stage 2 is designed to identify these concept nodes automatically under the assumption that most of them will have high relevancy rates. In other words, if we sort the concept nodes by relevancy rates then the domain-specific patterns should float to the top.

One of the side effects of this approach is that the statistics provide feedback on which heuristics are most appropriate. In previous work with AutoSlog, we found that some domains require longer extraction patterns than others [Riloff, 1994]. In particular, we found that simple verb forms usually suffice as extraction patterns in the terrorism domain (e.g., "X was killed"). But in the joint ventures domain, good extraction patterns often require both verbs and nouns (e.g., "X formed venture" is better than "X formed"). For this reason, we found it necessary to run AutoSlog with slightly different rule sets in these domains. In contrast, AutoSlog-TS simply allows all applicable heuristics to fire[8], often producing multiple extraction patterns of varying lengths, and lets the statistics ultimately decide which ones work the best. For example, "X formed" would presumably have a much lower relevancy rate than "X formed venture" in the joint ventures domain. The original version of AutoSlog could have applied multiple heuristics as well, but its dictionary had to be manually filtered so it was preferable to keep the dictionary small. Since AutoSlog-TS uses statistical filtering, we don't have to worry as much about the number of concept nodes generated and therefore don't need separate rule sets.

However, determining which concept nodes are ultimately "useful" depends on how one intends to use them. We are interested in using the concept nodes for two tasks: information extraction and text classification. These tasks place different demands on the concept node dictionary.

A good dictionary for information extraction should contain patterns that provide broad coverage of the domain. In general, useful patterns fall into one of two categories: (a) patterns that frequently extract relevant information and rarely extract irrelevant information or (b) patterns that frequently extract relevant information but often extract irrelevant information as well. Patterns of type (a) should have high relevancy rates. Patterns of type (b) are more difficult to identify but will occur with high frequency in relevant texts. Section 4.2 presents experiments with concept node filtering techniques for the information extraction task.

A good dictionary for text classification should contain patterns that frequently occur in relevant texts but rarely occur in irrelevant texts. These patterns represent expressions that are highly indicative of the domain and are therefore useful for classifying new texts. AutoSlog-TS was motivated by a text classification algorithm called the *relevancy signatures algorithm* [Riloff and Lehnert, 1994]. This algorithm applies CIRCUS to a preclassified training corpus and com-

[7]Most, but not all, noun phrases will yield a concept node. AutoSlog's heuristics sometimes fail to produce a concept node when the verb is weak (e.g., forms of "to be"), when the linguistic context does match any of the heuristics, or when CIRCUS produces a faulty sentence analysis.

[8]Referring back to Figure 3, heuristics 1 and 2 can fire in parallel, as can heuristics 1 and 4, and heuristics 8 and 9.

putes statistics to identify which *signatures* occur much more frequently in relevant texts than irrelevant texts (i.e., have a high relevancy rate). A signature consists of a concept node paired with the word that triggered it, although in the experiments presented in this paper there is a one-to-one correspondence between concept nodes and signatures.[9] The relevancy signatures algorithm essentially identifies concept nodes that have a high relevancy rate and uses them to classify new texts. Therefore, the AutoSlog-TS dictionary and statistics can be fed directly into the text classification algorithm. We present text classification results with AutoSlog-TS in the next section.

4 Experiments in the Terrorism Domain

We conducted a series of experiments with AutoSlog-TS to evaluate how well it performs on a text classification task, and to assess the viability of using it for information extraction tasks. First, we describe text classification results for the MUC-4 terrorism domain. Second, we present data that suggests how the dictionary can be filtered automatically for information extraction.

4.1 Text Classification Experiments

In the first experiment, we applied AutoSlog-TS to 1500 texts[10] from the MUC-4 corpus, which has been preclassified for the domain of Latin American terrorism. Roughly 50% of the texts are classified as relevant. AutoSlog-TS produced a dictionary of 32,345 unique concept nodes. To reduce the set of patterns down to a manageable size, we eliminated all concept nodes that were proposed exactly once, under the assumption that a pattern encountered only once is unlikely to be of much value. AutoSlog-TS often proposes the same pattern multiple times and keeps track of how often each pattern is proposed. After frequency filtering, the AutoSlog-TS dictionary contained 11,225 unique concept nodes.

We then ran CIRCUS over the same set of texts using the new concept node dictionary. For each text, we kept track of the concept nodes that were activated. We expect each concept node to be activated at least once, because these texts were used to create the concept node definitions.[11] This data was handed off to the relevancy signatures algorithm which generates signatures for each text (by pairing each concept node with the word that triggered it), and calculates statistics for each signature to identify how often it appeared in relevant texts versus irrelevant texts. The relevancy signatures algorithm uses a relevancy threshold R to identify the most relevant signatures and a frequency threshold M to eliminate signatures that were seen only a few times during training. Signatures that pass both thresholds are labeled as *relevancy signatures* and are used to classify new texts.

Finally, we evaluated the system by classifying two blind sets of 100 texts each, the TST3 and TST4 test sets from the MUC-4 corpus. Each new text was processed by CIRCUS and classified as relevant if it generated a relevancy signature. We compared these results with results produced

[9]The hand-crafted dictionary contains concept nodes that are triggered by multiple words but all of the concept nodes generated by AutoSlog are triggered by exactly one word.

[10]The DEV, TST1, and TST2 texts [MUC-4 Proceedings, 1992].

[11]A concept node may be activated by CIRCUS more often than it is proposed by AutoSlog-TS. For example, consider the phrase "the murder in Bogota by terrorists." To extract "terrorists", AutoSlog-TS uses a pp-attachment algorithm which should attach the PP to the noun "murder." However, it often makes mistakes and might attach the PP to the noun "Bogota." In this case, AutoSlog-TS would not propose the pattern "murder by X" even though it appears in the text.

by the hand-crafted MUC-4 dictionary. We ran each system 120 times using a variety of threshold settings: R was varied from 70 to 95 in increments of five, and M was varied from 1 to 20 in increments of one. Both text classification systems were trained on the same set of 1500 texts and were identical except that they used different concept node dictionaries. Figures 5 and 6 show the scatterplots.

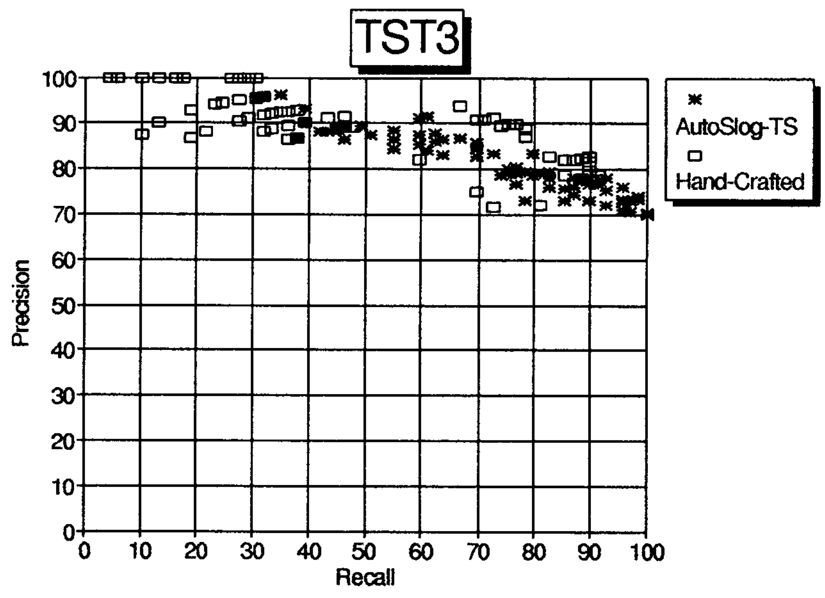

Figure 5: TST3 Text Classification Results for Different Dictionaries

The AutoSlog-TS dictionary performed comparably to the hand-crafted dictionary on both test sets. On TST4, the AutoSlog-TS dictionary actually achieved higher precision than the hand-crafted dictionary for recall levels $\leq$ 60%, and produced several data points that achieved 100% precision (the hand-crafted dictionary did not produce any). However, we see a trade-off at higher recall levels. The AutoSlog-TS dictionary achieved higher recall (up to 100%), which makes sense considering that the AutoSlog-TS dictionary is much bigger than the hand-crafted dictionary. But the hand-crafted dictionary achieved higher precision at recall levels above 60-65%. This is probably because the hand-crafted dictionary was filtered manually, which ensures that all of its concept nodes are relevant to the domain (although not all are useful as classifiers). In contrast, the AutoSlog-TS dictionary was not filtered manually so the statistics are solely responsible for separating the relevant concept nodes from the irrelevant ones. To achieve high recall, the threshold values must be low which allows some irrelevant patterns to pass threshold and cause erroneous classifications.

Overall, the text classification results from AutoSlog-TS are very encouraging. The AutoSlog-TS dictionary produced results comparable to a hand-crafted dictionary on both test sets and even surpassed the precision scores of the hand-crafted dictionary on TST4. Furthermore, the entire text classification system is constructed automatically using only a preclassified training corpus, and no text annotations or manual filtering of any kind.

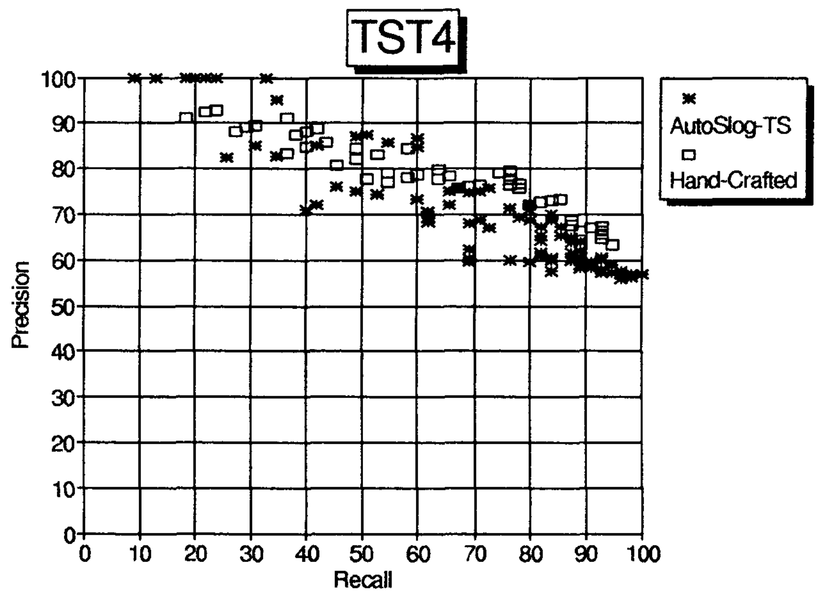

Figure 6: TST4 Text Classification Results with Different Dictionaries

4.2 Comparative Dictionary Experiments

We were also interested in gathering data to suggest how the AutoSlog-TS dictionary could be filtered automatically to produce an effective dictionary for information extraction. As we indicated in Section 3.2, a dictionary for text classification requires patterns that can discriminate between relevant and irrelevant texts. In contrast, a dictionary for information extraction requires patterns that will extract relevant information, but they may also extract irrelevant information. For example, in the terrorism domain, it is essential to have a pattern for the expression "X was killed" because people are frequently killed in terrorist attacks. However, this pattern is also likely to appear in texts that describe other types of incidents, such as accidents and military actions.

First, we collected data to compare the AutoSlog-TS dictionary with a dictionary produced by the original version of AutoSlog. The AutoSlog dictionary was generated using an annotated corpus and was subsequently filtered by a person, so it relied on two levels of human effort. The AutoSlog dictionary contains 428 unique concept node patterns[12], which were all deemed to be relevant by a person. The AutoSlog-TS dictionary contains 32,345 unique patterns of which 398 intersect with the AutoSlog dictionary.[13]

We experimented with automatic filtering techniques based on two criteria: frequency and relevancy. For frequency filtering, we simply removed all concept nodes that were proposed by AutoSlog-TS less than N times. For example, N=2 eliminated all concept nodes that were proposed exactly once and reduced the size of the dictionary from 32,345 to 11,225. Figure 7 shows the intersections between the AutoSlog-TS dictionary and the AutoSlog dictionary after frequency

[12]The dictionary actually contains 450 concept nodes but some concept nodes represent the same pattern to extract different types of objects. For example, the pattern "X was attacked" is used to extract both victims and physical targets.

[13]In theory, AutoSlog-TS should have generated all of the patterns that were generated by AutoSlog. However, AutoSlog-TS uses a slightly different version of CIRCUS and a new pp-attachment algorithm (see [Riloff, 1994]).

filtering. It is interesting to note that approximately half of the concept nodes in the AutoSlog dictionary were proposed fewer than 5 times by AutoSlog-TS. This implies that roughly half of the concept nodes in the AutoSlog dictionary occurred infrequently and probably had little impact on the overall performance of the information extraction system.[14] One of the problems with manual filtering is that it is difficult for a person to know whether a pattern will occur frequently or infrequently in future texts. As a result, people tend to retain many patterns that are not likely to be encountered very often.

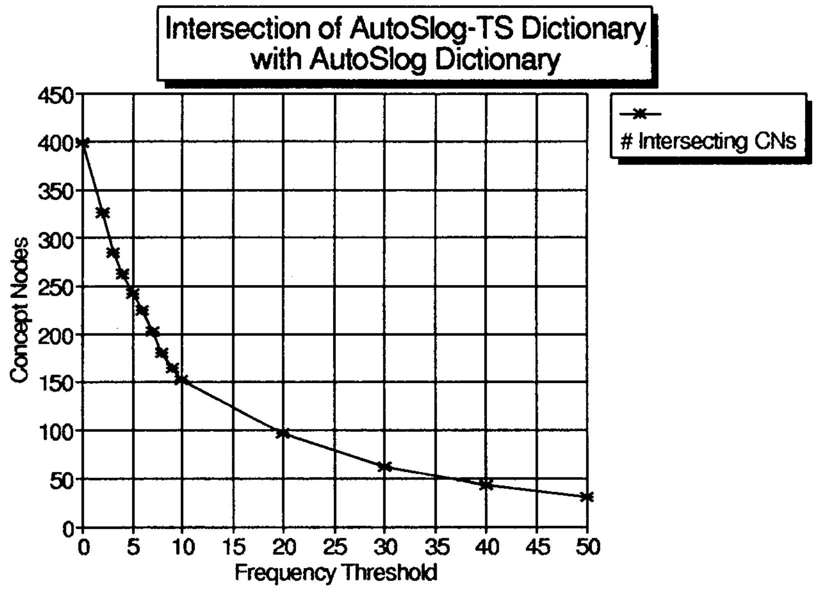

Figure 7: Comparison of Dictionaries Using Frequency Filtering

For relevancy filtering, we retained only the concept nodes that had $\geq$ N% correlation with relevant texts. For example, N=80 means that we retained a concept node if $\geq$ 80% of its occurrences were in relevant texts. Figure 8 shows the intersections between the dictionaries after relevancy filtering. Not surprisingly, most of the concept nodes in the AutoSlog dictionary had at least a 50% relevancy rate. However, the number of concept nodes drops off rapidly at higher relevancy rates. Again, this is not surprising because many useful extraction patterns will be common in both relevant and irrelevant texts.

Finally, we filtered the AutoSlog-TS dictionary using both relevancy and frequency filtering (N=5) to get a rough idea of how many concept node definitions will be useful for information extraction. Figure 9 shows the size of the resulting dictionaries after filtering. The number of concept nodes drops off dramatically from 32,345 to 4,169 after frequency filtering alone. There is a roughly linear relationship between the relevancy rate and the number of concept nodes retained.

It seems relatively safe to assume that concept nodes with a relevancy rate below 50% are not highly associated with the domain, and that concept nodes with a total frequency < 5 are probably not going to be encountered often. Using these two threshold values, we can reduce the size of the dictionary down to 1870 definitions. This dictionary is much more manageable in size

[14]This is consistent with earlier results which showed that a relatively small set of concept nodes typically do most of the work [Riloff, 1994].

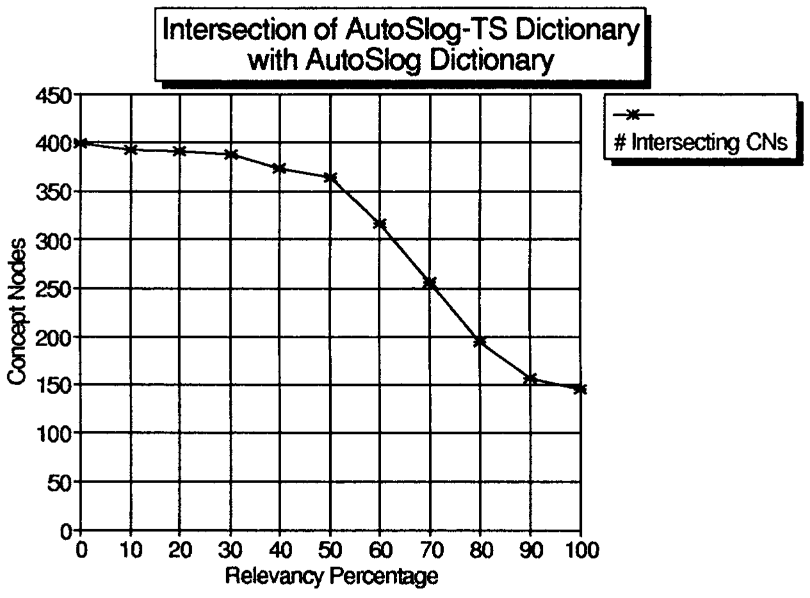

Figure 8: Comparison of Dictionaries Using Relevancy Filtering

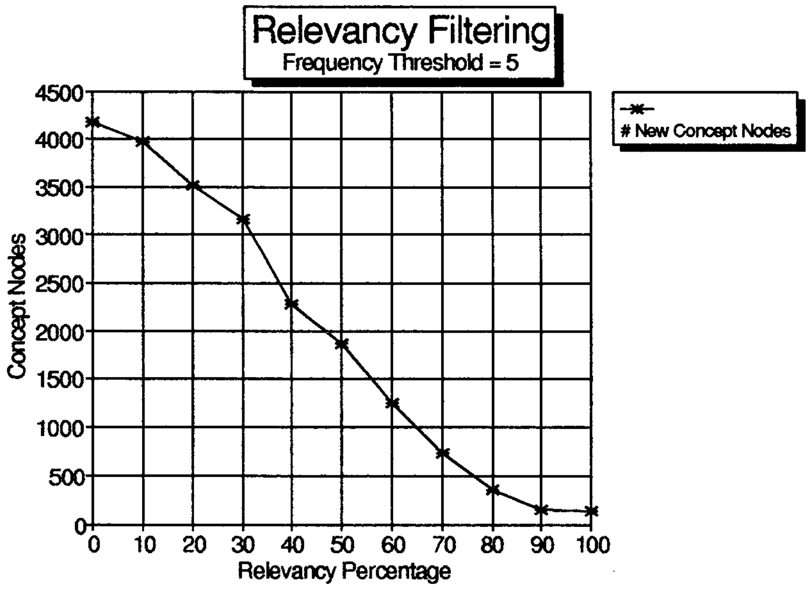

Figure 9: AutoSlog-TS Dictionary after Frequency and Relevancy Filtering

and could be easily reviewed by a person to separate the good definitions from the bad ones.[15] If for no other reason, a human would be required to assign semantic labels to each definition so that the system can identify the type of information that is extracted. Furthermore, the AutoSlog-TS dictionary should contain a higher percentage of relevant definitions that the original AutoSlog dictionary. Since the AutoSlog-TS dictionary has been prefiltered for both frequency and relevancy, many concept nodes that represent uncommon phrases or general expressions have already been removed.

Because AutoSlog-TS is not constrained to consider only the annotated portions of the corpus, it found many good patterns that AutoSlog did not. For example, AutoSlog-TS produced 158 concept nodes that have a relevancy rate $\geq 90\%$ and frequency ≥ 5. Only 45 of these concept nodes were in the original AutoSlog dictionary. Figure 10 shows a sample of some of the new concept nodes that represent patterns associated with terrorism.[16]

was assassinated in X	assassination in X	X ordered assassination
was captured by X	capture of X	X managed to escape
was exploded in X	damage in X	X expressed solidarity
was injured by X	headquarters of X	perpetrated on X
was kidnapped in X	targets of X	hurled at X
was perpetrated on X	went_off on X	carried_out X
was shot in X	X blamed	suspected X
was shot_to_death on X	X defused	to protest X
X was hit	X injured	to arrest X

Figure 10: Patterns found by AutoSlog-TS but not by AutoSlog

These results suggest that combining domain-independent linguistic rules with simple filtering techniques is a promising approach for automatically creating dictionaries of extraction patterns. Although it may still be necessary for a human to review the resulting patterns to build an information extraction system, this approach eliminates the need for text annotations and relies only on preclassified texts.

5 Discussion

AutoSlog-TS demonstrates that conceptual patterns for information extraction can be acquired automatically from only a preclassified text corpus, thereby obviating the need for an annotated training corpus. Generating annotated corpora is time-consuming and sometimes difficult, though the payoffs are often significant. General purpose text annotations, such as part-of-speech tags and noun-phrase bracketing, are costly to obtain but have wide applicability and have been used successfully to develop statistical NLP systems (e.g., [Church, 1989; Weischedel *et al.*, 1993]). Domain-specific text annotations, however, require a domain expert and have much narrower applicability.

From a practical perspective, it is important to consider the human factor and to try to minimize the amount of time and effort required to build a training corpus. Domain-specific text annotations are expensive to obtain, so our goal has been to eliminate our dependence on them.

[15] As we stated in Section 3.1, it took a person only 5 hours to review the 1237 concept nodes produced by AutoSlog [Riloff, 1993].

[16] The connected words represent phrases in CIRCUS' lexicon.

We have shown that a more coarse level of manual effort is sufficient for certain tasks. We have shown how a preclassified training corpus can be combined with statistical techniques to create conceptual patterns automatically. We believe that it is much easier for a person to separate a set of texts into two piles (the relevant texts and the irrelevant texts) than to generate detailed text annotations for a domain. Furthermore, the classifications are general in nature so various types of systems can make use of them. AutoSlog-TS suggests promising directions for future research in developing dictionaries automatically using only preclassified corpora without detailed text annotations.

References

Carbonell, J. G. 1979. Towards a Self-Extending Parser. In *Proceedings of the 17th Meeting of the Association for Computational Linguistics.* 3–7.

Church, K. 1989. A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. In *Proceedings of the Second Conference on Applied Natural Language Processing.*

Francis, W. and Kucera, H. 1982. *Frequency Analysis of English Usage.* Houghton Mifflin, Boston, MA.

Granger, R. H. 1977. FOUL-UP: A Program that Figures Out Meanings of Words from Context. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence.* 172–178.

Jacobs, P. and Zernik, U. 1988. Acquiring Lexical Knowledge from Text: A Case Study. In *Proceedings of the Seventh National Conference on Artificial Intelligence.* 739–744.

Kim, J. and Moldovan, D. 1993. Acquisition of Semantic Patterns for Information Extraction from Corpora. In *Proceedings of the Ninth IEEE Conference on Artificial Intelligence for Applications*, Los Alamitos, CA. IEEE Computer Society Press. 171–176.

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 Fourth Message Understanding Conference (MUC-4)*, San Mateo, CA. Morgan Kaufmann. 282–288.

Lehnert, W. 1991. Symbolic/Subsymbolic Sentence Analysis: Exploiting the Best of Two Worlds. In Barnden, J. and Pollack, J., editors 1991, *Advances in Connectionist and Neural Computation Theory, Vol. 1.* Ablex Publishers, Norwood, NJ. 135–164.

Marcus, M.; Santorini, B.; and Marcinkiewicz, M. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics* 19(2):313–330.

Proceedings of the Third Message Understanding Conference (MUC-3), San Mateo, CA. Morgan Kaufmann.

Proceedings of the Fourth Message Understanding Conference (MUC-4), San Mateo, CA. Morgan Kaufmann.

Proceedings of the Fifth Message Understanding Conference (MUC-5), San Francisco, CA. Morgan Kaufmann.

Riloff, E. and Lehnert, W. 1994. Information Extraction as a Basis for High-Precision Text Classification. *ACM Transactions on Information Systems* 12(3):296–333.

Riloff, E. 1993. Automatically Constructing a Dictionary for Information Extraction Tasks. In *Proceedings of the Eleventh National Conference on Artificial Intelligence.* AAAI Press/The MIT Press. 811–816.

Riloff, E. 1994. *Information Extraction as a Basis for Portable Text Classification Systems.* Ph.D. Dissertation, Department of Computer Science, University of Massachusetts Amherst.

Weischedel, R.; Meteer, M.; Schwartz, R.; Ramshaw, L.; and Palmucci, J. 1993. Coping with Ambiguity and Unknown Words through Probabilistic Models. *Computational Linguistics* 19(2):359–382.