In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pp. 1044-1049

Automatically Generating Extraction Patterns from Untagged Text

Ellen Riloff
Department of Computer Science
University of Utah
Salt Lake City, UT 84112
riloff@cs.utah.edu

Abstract

Many corpus-based natural language processing systems rely on text corpora that have been manually annotated with syntactic or semantic tags. In particular, all previous dictionary construction systems for information extraction have used an annotated training corpus or some form of annotated input. We have developed a system called AutoSlog-TS that creates dictionaries of extraction patterns using only untagged text. AutoSlog-TS is based on the AutoSlog system, which generated extraction patterns using annotated text and a set of heuristic rules. By adapting AutoSlog and combining it with statistical techniques, we eliminated its dependency on tagged text. In experiments with the MUC-4 terrorism domain, AutoSlog-TS created a dictionary of extraction patterns that performed comparably to a dictionary created by AutoSlog, using only preclassified texts as input.

Motivation

The vast amount of text becoming available on-line offers new possibilities for conquering the knowledge-engineering bottleneck lurking underneath most natural language processing (NLP) systems. Most corpus-based systems rely on a text corpus that has been manually tagged in some way. For example, the Brown corpus (Francis & Kucera 1982) and the Penn Treebank corpus (Marcus, Santorini, & Marcinkiewicz 1993) are widely used because they have been manually annotated with part-of-speech and syntactic bracketing information. Part-of-speech tagging and syntactic bracketing are relatively general in nature, so these corpora can be used by different natural language processing systems and for different domains. But some corpus-based systems rely on a text corpus that has been manually tagged in a domain-specific or task-specific manner. For example, corpus-based approaches to information extraction generally rely on special domain-specific text annotations. Consequently, the manual tagging effort is considerably less cost effective because the annotated corpus is useful for only one type of NLP system and for only one domain.

Corpus-based approaches to information extraction have demonstrated a significant time savings over conventional hand-coding methods (Riloff 1993). But the time required to annotate a training corpus is a non-trivial expense. To further reduce this knowledge-engineering bottleneck, we have developed a system called AutoSlog-TS that generates extraction patterns using untagged text. AutoSlog-TS needs only a *pre-classified* corpus of relevant and irrelevant texts. Nothing inside the texts needs to be tagged in any way.

Generating Extraction Patterns from Tagged Text

Related work

In the last few years, several systems have been developed to generate patterns for information extraction automatically. All of the previous systems depend on manually tagged training data of some sort. One of the first dictionary construction systems was AutoSlog (Riloff 1993), which requires tagged noun phrases in the form of annotated text or text with associated answer keys. PALKA (Kim & Moldovan 1993) is similar in spirit to AutoSlog, but requires manually defined frames (including keywords), a semantic hierarchy, and an associated lexicon. Competing hypotheses are resolved by referring to manually encoded answer keys, if available, or by asking a user.

CRYSTAL (Soderland *et al.* 1995) also generates extraction patterns using an annotated training corpus. CRYSTAL relies on both domain-specific annotations plus a semantic hierarchy and associated lexicon. LIEP (Huffman 1996) is another system that learns extraction patterns but relies on predefined keywords, object recognizers (e.g., to identify people and companies), and human interaction to annotate each relevant sentence with an event type. Cardie (Cardie 1993) and Hastings (Hastings & Lytinen 1994) also developed lexical acquisition systems for information extraction, but their systems learned individual word

meanings rather than extraction patterns. Both systems used a semantic hierarchy and sentence contexts to learn the meanings of unknown words.

AutoSlog

AutoSlog (Riloff 1996) is a dictionary construction system that creates extraction patterns automatically using heuristic rules. As input, AutoSlog needs answer keys or text in which the noun phrases that should be extracted have been labeled with domain-specific tags. For example, in a terrorism domain, noun phrases that refer to perpetrators, targets, and victims may be tagged. Given a tagged noun phrase and the original source text, AutoSlog first identifies the sentence in which the noun phrase appears. If there is more than one such sentence and the annotation does not indicate which one is appropriate, then AutoSlog chooses the first one. AutoSlog invokes a sentence analyzer called CIRCUS (Lehnert 1991) to identify clause boundaries and syntactic constituents. AutoSlog needs only a flat syntactic analysis that recognizes the subject, verb, direct object, and prepositional phrases of each clause, so almost any parser could be used. AutoSlog determines which clause contains the targeted noun phrase and applies the heuristic rules shown in Figure 1.

PATTERN	EXAMPLE
<subj> passive-verb	<victim> was murdered
<subj> active-verb	<perp> bombed
<subj> verb infin.	<perp> attempted to kill
<subj> aux noun	<victim> was victim
passive-verb <dobj>[1]	killed <victim>
active-verb <dobj>	bombed <target>
infin. <dobj>	to kill <victim>
verb infin. <dobj>	tried to attack <target>
gerund <dobj>	killing <victim>
noun aux <dobj>	fatality was <victim>
noun prep <np>	bomb against <target>
active-verb prep <np>	killed with <instrument>
passive-verb prep <np>	was aimed at <target>

Figure 1: AutoSlog Heuristics

The rules are divided into three categories, based on the syntactic class of the noun phrase. For example, if the targeted noun phrase is the subject of a clause, then the subject rules apply. Each rule generates an expression that likely defines the conceptual role of the noun phrase. In most cases, they assume that the verb determines the role. The rules recognize several verb forms, such as active, passive, and infinitive. An extraction pattern is created by instantiating the rule with the specific words that it matched in the sentence. The rules are ordered so the first one that is satisfied generates an extraction pattern, with the longer patterns being tested before the shorter ones. As an example, consider the following sentence:

```
Ricardo Castellar, the mayor, was kidnapped
yesterday by the FMLN.
```

Suppose that "Ricardo Castellar" was tagged as a relevant victim. AutoSlog passes the sentence to CIRCUS, which identifies Ricardo Castellar as the subject. AutoSlog's subject heuristics are tested and the **<subj> passive-verb** rule fires. This pattern is instantiated with the specific words in the sentence to produce the extraction pattern **<victim> was kidnapped**. In future texts, this pattern will be activated whenever the verb "kidnapped" appears in a passive construction, and its subject will be extracted as a victim.

AutoSlog can produce undesirable patterns for a variety of reasons, including faulty sentence analysis, incorrect pp-attachment, or insufficient context. Therefore a person must manually inspect each extraction pattern and decide which ones should be accepted and which ones should be rejected. This manual filtering process is typically very fast. In experiments with the MUC-4 terrorism domain, it took a user only 5 hours to review 1237 extraction patterns (Riloff 1993). Although this manual filtering process is part of the knowledge-engineering cycle, generating the annotated training corpus is a much more substantial bottleneck.

Generating Extraction Patterns from Untagged Text

To tag or not to tag?

Generating an annotated training corpus is a significant undertaking, both in time and difficulty. Previous experiments with AutoSlog suggested that it took a user about 8 hours to annotate 160 texts (Riloff 1996). Therefore it would take roughly a week to construct a training corpus of 1000 texts. Committing a domain expert to a knowledge-engineering project for a week is prohibitive for most short-term applications.

Furthermore, the annotation task is deceptively complex. For AutoSlog, the user must annotate relevant noun phrases. But what constitutes a relevant noun phrase? Should the user include modifiers or just the head noun? All modifiers or just the relevant modifiers? Determiners? If the noun phrase is part of a conjunction, should the user annotate all conjuncts or just one? Should the user include appositives? How about prepositional phrases? The meaning of simple NPs can change substantially when a prepositional phrase is at-

[1] In principle, passive verbs should not have direct objects. We included this pattern only because CIRCUS occasionally confused active and passive constructions.

tached. For example, "the Bank of Boston" is different from "the Bank of Toronto." Real texts are loaded with complex noun phrases that often include a variety of these constructs in a single reference. There is also the question of which references to tag. Should the user tag all references to a person? If not, which ones? It is difficult to specify a convention that reliably captures the desired information, but not specifying a convention can produce inconsistencies in the data.

To avoid these problems, we have developed a new version of AutoSlog, called AutoSlog-TS, that does not require any text annotations. AutoSlog-TS requires only a preclassified training corpus of relevant and irrelevant texts for the domain.[2] A preclassified corpus is much easier to generate, since the user simply needs to identify relevant and irrelevant sample texts. Furthermore, relevant texts are already available on-line for many applications and could be easily exploited to create a training corpus for AutoSlog-TS.

AutoSlog-TS

AutoSlog-TS is an extension of AutoSlog that operates exhaustively by generating an extraction pattern for every noun phrase in the training corpus. It then evaluates the extraction patterns by processing the corpus a second time and generating relevance statistics for each pattern. The process is illustrated in Figure 2.

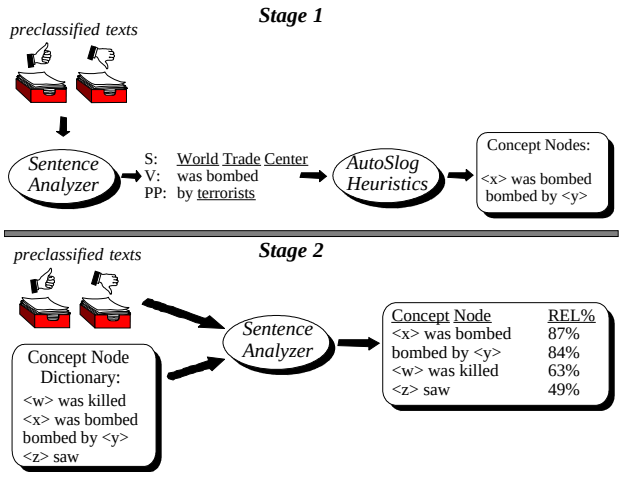

Figure 2: AutoSlog-TS flowchart

In Stage 1, the sentence analyzer produces a syntactic analysis for each sentence and identifies the noun phrases. For each noun phrase, the heuristic rules generate a pattern (called a *concept node* in CIRCUS) to extract the noun phrase. AutoSlog-TS uses a set of 15 heuristic rules: the original 13 rules used by AutoSlog plus two more: **<subj> active-verb dobj** and **infinitive prep <np>**. The two additional rules were created for a business domain from a previous experiment and are probably not very important for the experiments described in this paper.[3] A more significant difference is that AutoSlog-TS allows multiple rules to fire if more than one matches the context. As a result, multiple extraction patterns may be generated in response to a single noun phrase. For example, the sentence "terrorists bombed the U.S. embassy" might produce two patterns to extract the terrorists: **<subj> bombed** and **<subj> bombed embassy**. The statistics will later reveal whether the shorter, more general pattern is good enough or whether the longer pattern is needed to be reliable for the domain. At the end of Stage 1, we have a giant dictionary of extraction patterns that are literally capable of extracting every noun phrase in the corpus.

In Stage 2, we process the training corpus a second time using the new extraction patterns. The sentence analyzer activates all patterns that are applicable in each sentence. We then compute relevance statistics for each pattern. More specifically, we estimate the conditional probability that a text is relevant given that it activates a particular extraction pattern. The formula is:

$$\Pr(relevant\ text \mid text\ contains\ pattern_i) = \frac{rel-freq_i}{total-freq_i}$$

where $rel-freq_i$ is the number of instances of $pattern_i$ that were activated in relevant texts, and $total-freq_i$ is the total number of instances of $pattern_i$ that were activated in the training corpus. For the sake of simplicity, we will refer to this probability as a pattern's *relevance rate*. Note that many patterns will be activated in relevant texts even though they are not domain-specific. For example, general phrases such as "was reported" will appear in all sorts of texts. The motivation behind the conditional probability estimate is that domain-specific expressions will appear substantially more often in relevant texts than irrelevant texts.

Next, we rank the patterns in order of importance to the domain. AutoSlog-TS's exhaustive approach to pattern generation can easily produce tens of thousands of extraction patterns and we cannot reasonably expect a human to review them all. Therefore, we use a ranking function to order them so that a person only needs to review the most highly ranked patterns.

We rank the extraction patterns according to the formula: $relevance\ rate * log_2(frequency)$, unless the relevance rate is ≤ 0.5 in which case the function returns zero because the pattern is negatively correlated

[2] Ideally, the irrelevant texts should be "near-miss" texts that are similar to the relevant texts.

[3] See (Riloff 1996) for a more detailed explanation.

with the domain (assuming the corpus is 50% relevant). This formula promotes patterns that have a high relevance rate or a high frequency. It is important for high frequency patterns to be considered even if their relevance rate is only moderate (say 70%) because of expressions like "was killed" which occur frequently in both relevant and irrelevant texts. If only the patterns with the highest relevance rates were promoted, then crucial expressions like this would be buried in the ranked list. We do not claim that this particular ranking function is the best - to the contrary, we will argue later that a better function is needed. But this function worked reasonably well in our experiments.

Experimental Results

Automated scoring programs were developed to evaluate information extraction (IE) systems for the message understanding conferences, but the credit assignment problem for any individual component is virtually impossible using only the scores produced by these programs. Therefore, we evaluated AutoSlog and AutoSlog-TS by manually inspecting the performance of their dictionaries in the MUC-4 terrorism domain. We used the MUC-4 texts as input and the MUC-4 answer keys as the basis for judging "correct" output (MUC-4 Proceedings 1992).

The AutoSlog dictionary was constructed using the 772 relevant MUC-4 texts and their associated answer keys. AutoSlog produced 1237 extraction patterns, which were manually filtered in about 5 person-hours. The final AutoSlog dictionary contained 450 extraction patterns. The AutoSlog-TS dictionary was constructed using the 1500 MUC-4 development texts, of which about 50% are relevant. AutoSlog-TS generated 32,345 unique extraction patterns. To make the size of the dictionary more manageable, we discarded patterns that were proposed only once under the assumption that they were not likely to be of much value. This reduced the size of the dictionary down to 11,225 extraction patterns. We loaded the dictionary into CIRCUS, reprocessed the corpus, and computed the relevance rate of each pattern. Finally, we ranked all 11,225 patterns using the ranking function. The 25 top-ranked extraction patterns appear in Figure 3. Most of these patterns are clearly associated with terrorism, so the ranking function appears to be doing a good job of pulling the domain-specific patterns up to the top.

The ranked extraction patterns were then presented to a user for manual review.[4] The review process consists of deciding whether a pattern should be accepted or rejected, and labeling the accepted patterns.[5] For example, the second pattern **murder of <np>** was accepted and labeled as a murder pattern that will extract victims. The user reviewed the top 1970 patterns in about 85 minutes and then stopped because few patterns were being accepted at that point. In total, 210 extraction patterns were retained for the final dictionary. The review time was much faster than for AutoSlog, largely because the ranking scheme clustered the best patterns near the top so the retention rate dropped quickly.

1. <subj> exploded
2. murder of <np>
3. assassination of <np>
4. <subj> was killed
5. <subj> was kidnapped
6. attack on <np>
7. <subj> was injured
8. exploded in <np>
9. death of <np>
10. <subj> took_place
11. caused <dobj>
12. claimed <dobj>
13. <subj> was wounded
14. <subj> occurred
15. <subj> was located
16. took_place on <np>
17. responsibility for <np>
18. occurred on <np>
19. was wounded in <np>
20. destroyed <dobj>
21. <subj> was murdered
22. one of <np>
23. <subj> kidnapped
24. exploded on <np>
25. <subj> died

Figure 3: The Top 25 Extraction Patterns

Note that some of the patterns in Figure 3 were not accepted for the dictionary even though they are associated with terrorism. Only patterns useful for extracting perpetrators, victims, targets, and weapons were kept. For example, the pattern **exploded in <np>** was rejected because it would extract locations.

To evaluate the two dictionaries, we chose 100 blind texts from the MUC-4 test set. We used 25 relevant texts and 25 irrelevant texts from the TST3 test set, plus 25 relevant texts and 25 irrelevant texts from the TST4 test set. We ran CIRCUS on these 100 texts, first using the AutoSlog dictionary and then using the AutoSlog-TS dictionary. The underlying information extraction system was otherwise identical.

We scored the output by assigning each extracted item to one of four categories: *correct*, *mislabeled*, *duplicate*, or *spurious*. An item was scored as *correct* if it matched against the answer keys. An item was *mislabeled* if it matched against the answer keys but was extracted as the wrong type of object. For example, if "Hector Colindres" was listed as a murder victim but was extracted as a physical target. An item was a *duplicate* if it was coreferent with an item in the answer keys. For example, if "him" was extracted and coreferent with "Hector Colindres." The extraction pattern acted correctly in this case, but the extracted information was not specific enough. Correct items extracted more than once were also scored as duplicates. An item

[4] The author did the manual review for this experiment.

[5] Note that AutoSlog's patterns were labeled automatically by referring to the text annotations.

was *spurious* if it did not refer to any object in the answer keys. All items extracted from irrelevant texts were spurious. Finally, items in the answer keys that were not extracted were counted as *missing*. Therefore *correct* + *missing* should equal the total number of items in the answer keys.[6]

Tables 1 and 2 show the numbers obtained after manually judging the output of the dictionaries. We scored three items: perpetrators, victims, and targets. The performance of the two dictionaries was very similar. The AutoSlog dictionary extracted slightly more correct items, but the AutoSlog-TS dictionary extracted fewer spurious items.[7]

Slot	Corr.	Miss.	Mislab.	Dup.	Spur.
Perp	36	22	1	11	129
Victim	41	24	7	18	113
Target	39	19	8	18	108
Total	116	65	16	47	350

Table 1: AutoSlog Results

Slot	Corr.	Miss.	Mislab.	Dup.	Spur.
Perp	30	27	2	12	97
Victim	40	25	7	19	85
Target	32	23	17	16	58
Total	102	75	26	47	240

Table 2: AutoSlog-TS Results

We applied a well-known statistical technique, the two-sample t test, to determine whether the differences between the dictionaries were statistically significant. We tested four data sets: *correct*, *correct* + *duplicate*, *missing*, and *spurious*. The t values for these sets were 1.1012, 1.1818, 0.1557, and 2.27 respectively. The *correct*, *correct* + *duplicate*, and *missing* data sets were not significantly different even at the $p < 0.20$ significance level. These results suggest that AutoSlog and AutoSlog-TS can extract relevant information with comparable performance. The *spurious* data, however, was significantly different at the $p < 0.05$ significance level. Therefore AutoSlog-TS was significantly more effective at reducing spurious extractions.

We applied three performance metrics to this raw data: **recall**, **precision**, and the **F-measure**. We calculated recall as *correct* / (*correct* + *missing*), and computed precision as (*correct* + *duplicate*) / (*correct* + *duplicate* + *mislabeled* + *spurious*). The F-measure (MUC-4 Proceedings 1992) combines recall and precision into a single value, in our case with equal weight.

As the raw data suggests, Table 3 shows that AutoSlog achieved slightly higher recall and AutoSlog-TS achieved higher precision. The F-measure scores were similar for both systems, but AutoSlog-TS obtained slightly higher F scores for victims and targets. Note that the AutoSlog-TS dictionary contained only 210 patterns, while the AutoSlog dictionary contained 450 patterns, so AutoSlog-TS achieved a comparable level of recall with a dictionary less than half the size.

	AutoSlog			AutoSlog-TS		
Slot	Recall	Prec.	F	Recall	Prec.	F
Perp	.62	.27	.38	.53	.30	.38
Victim	.63	.33	.43	.62	.39	.48
Target	.67	.33	.44	.58	.39	.47
Total	.64	.31	.42	.58	.36	.44

Table 3: Comparative Results

The AutoSlog precision results are substantially lower than those generated by the MUC-4 scoring program (Riloff 1993). There are several reasons for the difference. For one, the current experiments were done with a debilitated version of CIRCUS that did not process conjunctions or semantic features. Although AutoSlog does not use semantic features to create extraction patterns, they can be incorporated as selectional restrictions in the patterns. For example, extracted victims should satisfy a *human* constraint. Semantic features were not used in the current experiments for technical reasons, but undoubtedly would have improved the precision of both dictionaries. Also, the previously reported scores were based on the UMass/MUC-4 system, which included a discourse analyzer that used domain-specific rules to distinguish terrorist incidents from other events. CIRCUS was designed to extract potentially relevant information using only local context, under the assumption that a complete IE system would contain a discourse analyzer to make global decisions about relevance.

Behind the scenes

It is informative to look behind the scenes and try to understand why AutoSlog achieved slightly better recall and why AutoSlog-TS achieved better precision. Most of AutoSlog's additional recall came from low frequency patterns that were buried deep in AutoSlog-TS's ranked list. The main advantage of corpus-tagging is that the annotations provide guidance so the system can more easily hone in on the relevant expressions. Without corpus tagging, we are at the mercy of the ranking function. We believe that the ranking function did a good job of pulling the most impor-

[6] "Optional" items in the answer keys were scored as correct if extracted, but were never scored as missing.

[7] The difference in *mislabeled* items is an artifact of the human review process, not AutoSlog-TS.

tant patterns up to the top, but additional research is needed to recognize good low frequency patterns.

In fact, we have reason to believe that AutoSlog-TS is ultimately capable of producing better recall than AutoSlog because it generated many good patterns that AutoSlog did not. AutoSlog-TS produced 158 patterns with a relevance rate $\geq 90\%$ and frequency ≥ 5. Only 45 of these patterns were in the original AutoSlog dictionary.

The higher precision demonstrated by AutoSlog-TS is probably a result of the relevance statistics. For example, the AutoSlog dictionary contains an extraction pattern for the expression **<subj> admitted**, but this pattern was found to be negatively correlated with relevance (46%) by AutoSlog-TS. Some of AutoSlog's patterns looked good to the human reviewer, but were not in fact highly correlated with relevance.

In an ideal ranking scheme, the "heavy hitter" extraction patterns should float to the top so that the most important patterns (in terms of recall) are reviewed first. AutoSlog-TS was very successful in this regard. Almost 35% recall was achieved after reviewing only the first 50 extraction patterns! Almost 50% recall was achieved after reviewing about 300 patterns.

Future Directions

The previous results suggest that a core dictionary of extraction patterns can be created after reviewing only a few hundred patterns. The specific number of patterns that need to be reviewed will ultimately depend on the breadth of the domain and the desired performance levels. A potential problem with AutoSlog-TS is that there are undoubtedly many useful patterns buried deep in the ranked list, which cumulatively could have a substantial impact on performance. The current ranking scheme is biased towards encouraging high frequency patterns to float to the top, but a better ranking scheme might be able to balance these two needs more effectively. The precision of the extraction patterns could also be improved by adding semantic constraints and, in the long run, creating more complex extraction patterns.

AutoSlog-TS represents an important step towards making information extraction systems more easily portable across domains. AutoSlog-TS is the first system to generate domain-specific extraction patterns automatically without annotated training data. A user only needs to provide sample texts (relevant and irrelevant), and spend some time filtering and labeling the resulting extraction patterns. Fast dictionary construction also opens the door for IE technology to support other tasks, such as text classification (Riloff & Shoen 1995). Finally, AutoSlog-TS represents a new approach to exploiting on-line text corpora for domain-specific knowledge acquisition by squeezing preclassified texts for all they're worth.

Acknowledgments

This research was funded by NSF grant MIP-9023174 and NSF grant IRI-9509820. Thanks to Kem Mason and Jay Shoen for generating much of the data.

References

Cardie, C. 1993. A Case-Based Approach to Knowledge Acquisition for Domain-Specific Sentence Analysis. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, 798–803. AAAI Press/The MIT Press.

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

Hastings, P., and Lytinen, S. 1994. The Ups and Downs of Lexical Acquisition. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 754–759. AAAI Press/The MIT Press.

Huffman, S. 1996. Learning information extraction patterns from examples. In Wermter, S.; Riloff, E.; and Scheler, G., eds., *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*. Springer-Verlag, Berlin.

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*, 171–176. Los Alamitos, CA: IEEE Computer Society Press.

Lehnert, W. 1991. Symbolic/Subsymbolic Sentence Analysis: Exploiting the Best of Two Worlds. In Barnden, J., and Pollack, J., eds., *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.

MUC-4 Proceedings. 1992. *Proceedings of the Fourth Message Understanding Conference (MUC-4)*. San Mateo, CA: Morgan Kaufmann.

Riloff, E., and Shoen, J. 1995. Automatically Acquiring Conceptual Patterns Without an Annotated Corpus. In *Proceedings of the Third Workshop on Very Large Corpora*, 148–161.

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

Riloff, E. 1996. An Empirical Study of Automated Dictionary Construction for Information Extraction in Three Domains. *Artificial Intelligence*. Vol. 85. Forthcoming.

Soderland, S.; Fisher, D.; Aseltine, J.; and Lehnert, W. 1995. CRYSTAL: Inducing a conceptual dictionary. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 1314–1319.