Empirical Methods in Information Extraction

Claire Cardie
Department of Computer Science
Cornell University
Ithaca, NY 14850
E-mail: cardie@cs.cornell.edu

Most corpus-based methods in natural language processing (NLP) were developed to provide an arbitrary text-understanding application with one or more general-purpose linguistic capabilities. This is evident from the articles in this issue of *AI Magazine*. Charniak and Ng/Zelle, for example, describe techniques for part-of-speech tagging, parsing, and word-sense disambiguation. These techniques were created with no specific domain or high-level language-processing task in mind. In contrast, this article surveys the use of empirical methods for a *particular* natural language understanding task that is inherently *domain-specific*. The task is *information extraction*. Very generally, an information extraction system takes as input an unrestricted text and "summarizes" the text with respect to a prespecified topic or domain of interest: it finds useful information about the domain and encodes that information in a structured form, suitable for populating databases. In contrast to in-depth natural language understanding tasks, information extraction systems effectively skim a text to find relevant sections and then focus only on these sections in subsequent processing. The information extraction system in Figure 1, for example, summarizes stories about natural disasters, extracting for each such event the type of disaster, the date and time that it occurred, and data on any property damage or human injury caused by the event.

Information extraction has figured prominently in the field of empirical NLP: The first large-scale, head-to-head evaluations of NLP systems on the same text-understanding tasks were the DARPA-sponsored MUC[1] performance evaluations of information extraction systems (Lehnert and Sundheim, 1991; Chinchor *et al.*, 1993). Prior to each evaluation, all participating sites receive a corpus of texts from a predefined domain and the corresponding "answer keys" to use for system development. The answer keys are manually encoded templates — much like that of Figure 1 — that capture all information from the corresponding source text that is relevant to the domain, as specified in a set of written guidelines. After a short development phase[2], the NLP systems are evaluated by comparing the summaries each produces with the summaries generated by human experts for the same test set of previously unseen texts. The comparison is performed using an automated scoring program that rates each system according to measures of recall and precision. *Recall* measures the amount of the relevant information that the NLP system correctly extracts from the test collection while *precision* measures the reliability of the information extracted:

recall = (# correct slot-fillers in output template) / (# slot-fillers in answer key)
precision = (# correct slot-fillers in output template) / (# slot-fillers in output template)

As a result of MUC and other information extraction efforts, information extraction has become an increasingly viable technology for real-world text-processing applications. For example, there are currently information extraction systems that:

- Support underwriters in analyzing life insurance applications (Glasgow *et al.*, 1997).

[1] MUC is an acronym for Message Understanding Conference.
[2] The development phase has varied from year to year, but has ranged from one to nine months.

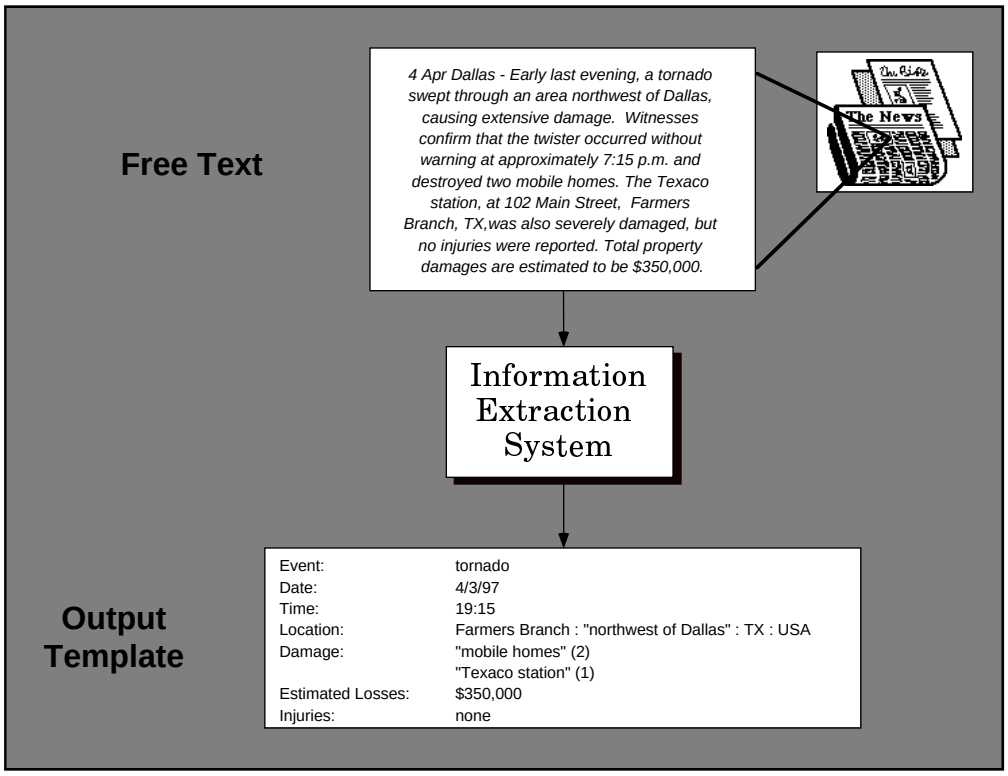

Figure 1: Information Extraction System in the Domain of Natural Disasters.

- Summarize medical patient records by extracting diagnoses, symptoms, physical findings, test results, and therapeutic treatments in order to assist health care providers or support insurance processing (Soderland *et al.*, 1995).
- Analyze newswires and transcripts of radio and television broadcasts to find and summarize descriptions of terrorist activities (MUC-3, 1991; MUC-4, 1992).
- Monitor technical articles describing microelectronic chip fabrication in order to capture information on chip sales, manufacturing advances, and the development or use of chip processing technologies (MUC-5, 1994).
- Analyze newspaper articles with the goal of finding and summarizing instances of business joint ventures (MUC-5, 1994).
- Support the automatic classification of legal documents (Holowczak and Adam, 1997).

There are also a growing number of Internet applications that employ information extraction technologies. Some examples include NLP systems that build knowledge bases directly from Web pages (Craven *et al.*, 1997); that create job listing databases from newsgroups, Web sites, and classified advertisements (see http://www.junglee.com/success/index.html); that build newsgroup query systems (Thompson *et al.*, 1997); and that create weather forecast databases from Web pages (Soderland, 1997).

While the MUC evaluations have shown that it is possible to rigorously evaluate some aspects of an information extraction system, it is difficult to state the overall performance levels of today's information extraction systems: at a minimum, performance depends on the relative complexity of the extraction task, the quality of the knowledge bases available to the NLP system, the syntactic and semantic complexity of the documents to be processed, and the regularity of the language in those documents. In general, however, the best extraction systems now can achieve levels of about 50%

recall and 70% precision on fairly complex information extraction tasks and can reach much higher levels of performance (approximately 90% recall and precision) for the easiest tasks. Although these levels of performance may not initially seem impressive, one should realize that information extraction is difficult for people as well as for machines. Will's [1993] study, for example, showed that the best machine extraction systems have an error rate that is only twice that of highly skilled analysts specifically trained in information extraction tasks.

In spite of this recent progress, there remain problems in today's information extraction systems. First, the accuracy and robustness of machine extraction systems can still be greatly improved. In particular, human error during information extraction is generally caused by a lapse of attention while the errors of an automated extraction system are due to its relatively shallow understanding of the input text. As a result, the machine-generated errors are more difficult to track down and to correct. Second, building an information extraction system in a new domain is difficult and time-consuming, often requiring months of effort by domain specialists and computational linguists familiar with the underlying NLP system. Part of the problem lies in the domain-specific nature of the task: an information extraction system will work better if its linguistic knowledge sources are tuned to the particular domain, but manually modifying and adding domain-specific linguistic knowledge to an existing NLP system is slow and error-prone.

The remainder of the article surveys the empirical methods in NLP that have been developed to address these problems of accuracy, portability, and knowledge acquisition for information extraction systems. Like the companion articles in this issue, we will see that empirical methods for information extraction are corpus-based, machine learning algorithms. To start, we present a generic architecture for information extraction systems. Next, we provide examples of the empirical methods designed to increase the accuracy or the portability of each component in the extraction system. Throughout we will focus on the specific needs and constraints that information extraction places on the language-learning tasks.

1 The Architecture of an Information Extraction System

In the early days of information extraction, NLP systems varied widely in their approach to the information extraction task. At one end of the spectrum were systems that processed a text using traditional NLP techniques: first a full syntactic analysis of each sentence, then semantic analysis of the resulting syntactic structures, and finally a discourse-level analysis of the syntactic and semantic representations. At the other extreme lie systems that used keyword matching techniques and little or no linguistic analysis of the input text. As more information extraction systems were built and empirically evaluated, however, researchers began to converge on a standard architecture for information extraction systems. That architecture is shown in Figure 2. Although many variations exist from system to system, the figure indicates the main functions performed in an information extraction system.

Each input text is first divided into sentences and words in a **Tokenization and Tagging** step. As indicated in Figure 2, many systems also disambiguate, or tag, each word with respect to part of speech and possibly semantic class at this point during processing. The **Sentence Analysis** phase follows. It comprises one or more stages of syntactic analysis, or parsing, that together identify noun groups, verb groups, prepositional phrases, and other simple constructs. In some systems, the parser also locates surface-level subjects and direct objects and identifies conjunctions, appositives, and other complex phrases. At some point, either before, during, or after the main steps of syntactic analysis, an information extraction system also finds and labels semantic entities relevant to the extraction topic. In the natural disaster domain, for example, the system might identify locations, company names, person names, time expressions, and money expressions, saving each in a normalized form.

Figure 2 shows the syntactic constituents and semantic entities identified during Sentence Analysis for the first sentence of the sample text. There are important differences between the Sentence Analysis stage of an information extraction system and traditional parsers. Most importantly, the goal of syntactic analysis in an information extraction system is **not** to produce a complete, detailed

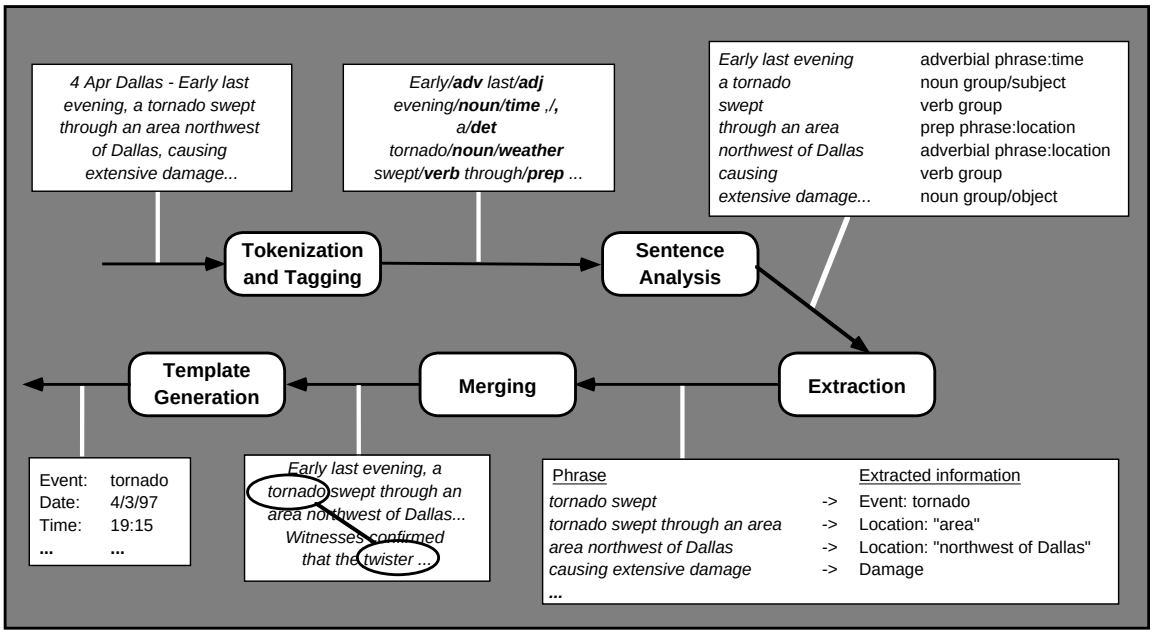

Figure 2: Architecture for an Information Extraction System.

parse tree for each sentence in the text. Instead, the system need only perform *partial parsing*: that is, it need only construct as much structure as the information extraction task requires. Unlike traditional full-sentence parsers, a partial parser looks for fragments of text that can be reliably recognized, e.g., noun groups and verb groups. Because of its limited coverage, a partial parser can rely solely on general pattern-matching techniques — often finite-state machines — to identify these fragments deterministically based on purely local syntactic cues. Partial parsing is well-suited for information extraction applications for an additional reason: the ambiguity resolution decisions that make full-blown parsing difficult can be postponed until later stages of processing where top-down expectations from the information extraction task can guide the system's actions.

The **Extraction** phase is the first entirely domain-specific component of the system. During Extraction, the system identifies domain-specific relations among relevant entities in the text. Given the first sentence in our example, this component should identify the type of natural disaster ("tornado"), the location of the event ("area" and "northwest of Dallas, TX"), and the fact that there was some property damage. Figure 2 shows the information extracted from the first sentence and those portions of text responsible for each piece of extracted data. Information for filling the remaining slots of the output template would be similarly extracted from subsequent sentences.

The main job of the **Merging** phase is coreference resolution, or anaphora resolution: the system examines each entity encountered in the text and determines whether it refers to an existing entity or whether it is new and must be added to the system's discourse-level representation of the text. In the sample text, for example, the mention of "a tornado" in the first sentence indicates a new entity; "the twister" in sentence two, however, refers to the same entity as the "tornado" of sentence one. Recognizing when two statements refer to the same entity is critical for an information extraction system because it allows the system to associate the information from both statements with the same object. In some systems, another task of Merging is to determine the implicit subjects of all verb phrases. In sentence one, this component would infer that "tornado" is the subject of "causing" (as well as the subject of "swept"), allowing the system to directly associate "damage" with the tornado. The discourse-level inferences made during Merging aid the **Template Generation** phase, which determines the number of distinct events in the text, maps the individually extracted pieces of information onto each event, and produces output templates. Purely domain-specific inferences may also occur during Template Generation. In the MUC terrorism domain, for example, terrorist

events involving only military targets were not considered relevant **unless** civilians were injured or there was damage to civilian property. The Template Generation phase is often the best place to apply this domain-specific constraint. In addition, some slots in the output template must be filled with terms chosen from a set of possibilities rather than a string from the input text. In the sample scenario, the Injuries and Event slots require such "set fills." Still other slots (e.g., Date, Time, Location) may require normalization of their fillers. Both of these subtasks are part of the Template Generation phase.

2 The Role of Corpus-Based Language Learning Algorithms

With this architecture in mind, we can now return to our original question: How have researchers used empirical methods in NLP to improve the accuracy and portability of information extraction systems? In general, corpus-based language learning algorithms have been used to improve individual *components* of the information extraction system and, as a result, to improve the end-to-end performance. In theory, empirical methods can be used for each subtask of information extraction: part-of-speech tagging, semantic class tagging, word-sense disambiguation, finding named entities (e.g., company names, person names, locations), partial parsing, learning extraction patterns, coreference resolution, and each step of template generation. The catch, as is often the case in corpus-based approaches to language learning, is obtaining enough training data. As described in the overview article, supervised language learning algorithms acquire a particular language-processing skill by taking many examples of how to correctly perform that task and then generalizing from the examples to handle unseen cases. The algorithms, therefore, critically depend on the existence of a corpus that has been annotated with the appropriate supervisory information. For language tasks that are primarily domain-independent and syntactic in nature, annotated corpora like the Penn Treebank (Marcus *et al.*, 1993) already exist and can be used to extract training data for the information extraction system. Part-of-speech tagging and bracketing text into noun groups, verb groups, clauses, etc., fall into this category. For these tasks, one can use the Treebank's Wall Street Journal corpus, which has been annotated with both word class and syntactic structure, together with any of a number of corpus-based algorithms, e.g., HMM's for part-of-speech tagging and statistical learning techniques for parsing (see (Charniak, 1993)); Brill's transformation-based learning for part-of-speech tagging (Brill, 1995) and bracketing (Ramshaw and Marcus, 1995); decision tree models for parsing (Magerman, 1995); case-based learning for lexical tagging (Cardie, 1993; Daelemans *et al.*, 1996), and inductive logic programming for learning syntactic parsers (Zelle and Mooney, 1994). The resulting taggers and bracketers will be effective across information extraction tasks as long as the input to the information extraction system employs a writing style and genre that is similar to the training corpus. Otherwise, a new training corpus must be created and used to completely retrain or to bootstrap the training of the component. In theory, word-sense disambiguation algorithms would also be portable across extraction tasks. But defining "standard" word senses is difficult and, to date, text collections have been annotated according to these predefined senses only for a small number of selected words. In addition, the importance of word-sense disambiguation for information extraction tasks remains unclear.

Natural language learning techniques are more difficult to apply to subsequent stages of information extraction — namely, learning extraction patterns, coreference resolution, and template generation. There are a number of problems. First, there are usually no corpora annotated with the appropriate semantic and domain-specific supervisory information. The typical "corpus" for information extraction tasks is a collection of texts and their associated answer keys, i.e., the output templates that should be produced for each text. This means that a new corpus must be created for each new information extraction task. In addition, the corpus simply does not contain the supervisory information needed to train most components of an information extraction system including the lexical tagging, coreference resolution, and template generation modules. The output templates are often inadequate even for learning extraction patterns: they indicate which strings should be extracted and how they should be labeled, but say nothing about which occurrence of the string is responsible for the extraction when multiple occurrences appear in the text. Furthermore, they

provide no direct means for learning patterns to extract set fills, symbols not necessarily appearing anywhere in the text. As a result, researchers create their own training corpora; but, because this process is slow, the resulting corpora may be much smaller than is normally required for statistical approaches to language analysis.

Another problem is that the semantic and domain-specific language-processing skills needed for information extraction often require the output of earlier levels of analysis, e.g., tagging and partial parsing. This complicates the generation of training examples for the learning algorithm because there can be no standard corpus from which complete training examples can be "read off" as is the case for part-of-speech tagging and parsing. The features that describe the learning problem depend on the information available to the extraction system in which the learning algorithm is embedded and these features become available only *after* the training texts have passed through earlier stages of linguistic analysis. Whenever the behavior of these earlier modules changes, new training examples must be generated and the learning algorithms for later stages of the information extraction system retrained. Furthermore, the learning algorithms must deal effectively with noise caused by errors from earlier components. The cumulative effect of the above complications is that the learning algorithms used for low-level tagging or syntactic analysis may not readily apply to the acquisition of these higher-level language skills and new algorithms often need to be developed.

In spite of the difficulties of applying empirical methods to problems in information extraction, it is precisely the data-driven nature of corpus-based approaches that allows them to simultaneously address both of the major problems in information extraction systems — accuracy and portability. When the training data is derived from the same type of texts that the information extraction system is to process, the acquired language skills are automatically tuned to that corpus, increasing the accuracy of the system. In addition, because each natural language understanding skill is learned automatically rather than being manually coded into the system, that skill can be moved quickly from one information extraction system to another by retraining the appropriate component.

The remaining sections describe the natural language learning techniques that have been developed for training the domain-dependent and semantic components of an information extraction system: Extraction, Merging, and Template Generation. In each case, we will describe how the above problems are addressed and summarize the state-of-the-art in the field.

3 Learning Extraction Patterns

As in the Sentence Analysis stages, general pattern-matching techniques have also become the technique of choice for the Extraction phase of an information extraction system. The role for empirical methods in the Extraction phase, therefore, is one of knowledge acquisition: to automate the acquisition of "good" extraction patterns where good patterns are patterns that are general enough to extract the correct information from more than one sentence, but specific enough so that they do not apply in inappropriate contexts. A number of researchers have investigated the use of corpus-based methods for learning information extraction patterns. The learning methods vary along a number of dimensions: the class of patterns learned, the training corpus required, the amount and type of human feedback required, the degree of preprocessing necessary, the background knowledge required, and the biases inherent in the learning algorithm itself.

One of the earliest systems for acquiring extraction patterns was the AutoSlog system (Lehnert *et al.*, 1992; Riloff, 1993). AutoSlog learns extraction patterns in the form of domain-specific "concept node" definitions for use with the CIRCUS parser (Lehnert, 1990; Cardie and Lehnert, 1991). AutoSlog's concept nodes can be viewed as domain-specific semantic case frames that contain a maximum of one slot per frame. Figure 3, for example, shows the concept node for extracting "two mobile homes" as damaged property from sentence two of the sample text. The first field in the concept node specifies the type of concept to be recognized (e.g., Damaged-Object). The concept type generally corresponds to a specific slot in the output template (e.g., the Damage slot of Figure 1). The remaining fields in the concept node represent the extraction pattern. The "Trigger" is the word that activates the pattern — it acts as the pattern's conceptual anchor point. "Position" denotes the syntactic position where the concept is expected to be found in the input sentence (e.g.,

Sentence Two: "Witnesses confirm that the twister occurred without warning at approximately 7:15 p.m and *destroyed two mobile homes.*"

Concept Node Definition:

Concept = Damaged-Object
Trigger = "destroyed"
Position = direct-object
Constraints = ((physical-object))
Enabling Conditions = ((active-voice))

Instantiated Concept Node

Damaged-Object = "two mobile homes"

Figure 3: Concept Node for Extracting "Damage" Information.

the direct object, subject, object of a preposition); the "Constraints" are selectional restrictions that apply to any potential instance of the concept. In CIRCUS, these semantic constraints can be hard or soft: Hard constraints are predicates that must be satisfied before the phrase in the specified Position can be extracted as an instance of the Concept; soft constraints suggest preferences for slot fillers, but do not inhibit the extraction of phrases if violated. In all of our examples, we will assume that the constraints are hard constraints. Finally, the "Enabling Conditions" are constraints on the linguistic context of the triggering word that must be satisfied before the pattern is activated. The concept node of Figure 3, for example, would be triggered by the word "destroyed" when used in the active voice. Once activated, the concept node would then extract the direct object of the clause to fill its Damaged-Object slot as long as that phrase denoted a physical object.

Once the concept node is defined, it can be used in conjunction with a partial parser to extract information from novel input sentences. The system parses the sentence; if the trigger word is encountered and the enabling conditions satisfied, then the phrase found in the specified syntactic constituent is extracted, tested for the appropriate semantic constraints, and then labeled as an instance of the designated concept type. The bottom of Figure 3 shows the concept extracted from sentence two of the sample text after applying the Damaged-Object concept node. Alternatively, given the sentence "The hurricane destroyed two office buildings," the same Damaged-Object concept node would extract "two office buildings" as the damaged entities. The extracted concepts are used during Merging and Template Generation to produce the desired output templates.

AutoSlog learns concept node definitions via a one-shot learning algorithm designed specifically for the information extraction task. As a training corpus, it requires a set of texts with noun phrases annotated with the appropriate concept type whenever they correspond to information that should be extracted from the text. In place of an annotated corpus, AutoSlog can also use a set of texts and their associated answer keys as provided in the MUC training corpus.[3] The AutoSlog learning algorithm is straightforward and depends only on the existence of a partial parser and a small set (approximately 13) of general linguistic patterns that direct the creation of concept nodes. The AutoSlog learning algorithm is straightforward. Given a noun phrase to be extracted, AutoSlog performs the following steps to derive a pattern for extracting the phrase:

1. *Find the sentence from which the noun phrase originated.* For example, given the target noun phrase "two mobile homes" marked as a Damaged-Object, AutoSlog would return sentence two from the sample text during this step.

2. *Present the sentence to the partial parser for processing.* AutoSlog's partial parser must be able to identify the subject, direct object, verb group, and prepositional phrases of each clause.

[3] A newer version of AutoSlog requires only that individual texts are marked as relevant or irrelevant to the domain (Riloff, 1996).

For sentence two of the sample text, the parser should determine, among other things, that "destroyed" occurred as the verb group of the third clause with "two mobile homes" as its direct object.

3. *Apply the linguistic patterns in order.* AutoSlog's linguistic patterns attempt to identify domain-specific thematic role information for a target noun phrase based on the syntactic position in which the noun phrase appears and the local linguistic context. The first pattern that applies determines the extraction pattern, i.e., concept node, for extracting the noun phrase from the training sentence. The linguistic pattern that would apply in the "two mobile homes" example is the following:

   <active-voice-verb> followed by <target-np>=<direct object>

   This pattern says that the noun phrase to be extracted, i.e., the target-np, appeared as the direct object of an active voice verb. Similar patterns exist for direct objects of passives and infinitives, and for cases where the target noun phrase appears as the subject of a clause or the object of a prepositional phrase. AutoSlog's linguistic patterns are for the most part domain-independent; they need little or no modification when moving an NLP system from one information extraction task to another.

4. *When a pattern applies, generate a concept node definition from the matched constituents, their context, the concept type provided in the annotation for the target noun phrase, and the predefined semantic class for the filler.* From the linguistic pattern shown above and the Damaged-Object annotation from sentence two, AutoSlog would generate a concept node definition of the following form:

   Concept = < <concept> of <target-np> >
   Trigger = "< <verb> of <active-voice-verb> >"
   Position = direct-object
   Constraints = ((< <semantic class> of <concept> >))
   Enabling Conditions = ((active-voice))

   AutoSlog assumes that the semantic class of each concept type is given as part of the domain specification and that the parser has a mechanism for assigning these semantic classes to nouns and noun modifiers during sentence analysis. After substitutions, this concept node definition will match the Damage concept node of Figure 3.

Some examples of extraction patterns learned by AutoSlog for the terrorism domain include (in shorthand form): <victim> was **murdered**; <perpetrator> **bombed**; <perpetrator> attempted to **kill**; was **aimed** at <target>. In these examples, the bracketed items denote concept type and the word in boldface is the concept node trigger. Although many of AutoSlog's learned patterns are good, some are too general (e.g., they are triggered by "is" or "are"); others are too specific; still others are just wrong. These "bad" extraction patterns are sometimes caused by parsing errors; alternatively, they occur when target noun phrases occur in a prepositional phrase and AutoSlog cannot determine whether the preceding verb or noun phrase should trigger the extraction. As a result, AutoSlog requires that a person review the proposed extraction patterns and discard those that seem troublesome.

AutoSlog has been used to automatically derive extraction patterns for a number of domains: terrorism, business joint ventures, and advances in microelectronics. In terms of improving the portability of information extraction systems, AutoSlog allowed developers to create extraction patterns for the terrorism domain in 5 hours instead of the approximately 1200–1500 hours required to create a set of extraction patterns for the domain by hand. In terms of accuracy, there was no direct empirical evaluation of the learned patterns even though some form of cross-validation could have been used. Instead, the learned patterns were evaluated indirectly — by using them in the

UMass/MUC-3 information extraction system and then measuring the overall performance of the information extraction system. The AutoSlog-generated patterns achieved 98% of the performance of the handcrafted patterns. This is especially impressive because the UMass system earned the highest combined recall and precision scores in the MUC-3 performance evaluation (Lehnert *et al.*, 1991). AutoSlog offered an additional advantage over the handcrafted rule set: because domain experts can review the automatically generated extraction patterns with minimal training, building the patterns no longer required the expertise of a computational linguist with a deep understanding of the underlying NLP system. This is a critical step towards building information extraction systems that are trainable entirely by end-users.

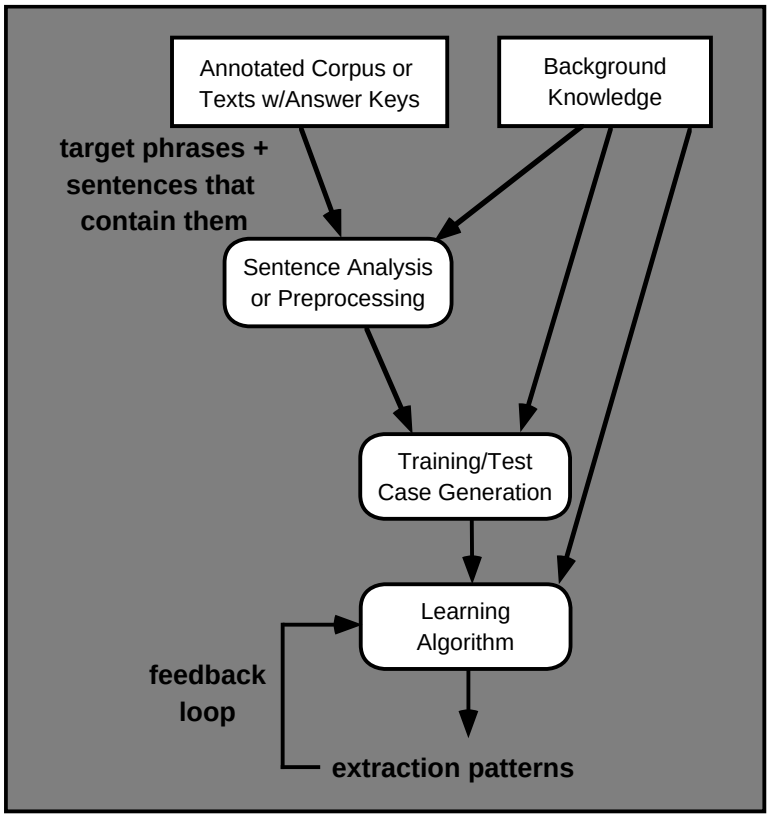

Figure 4: Learning Information Extraction Patterns.

Figure 4 shows the general structure of corpus-based approaches to learning information extraction patterns. AutoSlog conforms to this structure except for its human feedback loop, which filters bad extraction patterns but does not, in turn, inform the learning algorithm of its findings. Virtually all subsequent attempts to automate the acquisition of extraction patterns also conform to the general structure of Figure 4. In the next paragraphs, we describe a handful of these systems.

First, Kim and Moldovan's PALKA system [1995] learns extraction patterns that are similar in form to AutoSlog's concept nodes. The approach used to generate the patterns, however, is quite different. The background knowledge is not a set of linguistic patterns to be instantiated, but a concept hierarchy, a set of predefined keywords that can be used to trigger each pattern, and a semantic class lexicon. The concept hierarchy comprises a set of generic semantic case frame definitions for each type of information to be extracted. To learn extraction patterns, PALKA looks for sentences containing the keywords, parses those sentences, and then maps phrases in the parsed sentence to case frame slots using semantic class information. The training corpus is used to choose the correct mapping when more than one are possible while the concept and semantic class hierarchies guide PALKA's generalization and specialization of proposed patterns.

Like AutoSlog and PALKA, the CRYSTAL system (Soderland *et al.*, 1995) learns extraction

patterns in the form of semantic case frames. CRYSTAL's patterns, however, can be more complicated. Instead of specifying a single trigger word and its local linguistic context (i.e., the Enabling Conditions of Figure 3), the "triggers" for CRYSTAL's patterns comprise a much more detailed specification of linguistic context. In particular, the target constituent or any surrounding constituents (e.g., the subject, verb, or object of the current clause) may be tested for a specific sequence of words or for the presence of heads or modifiers with the appropriate semantic class. CRYSTAL employs a covering algorithm to learn extraction patterns and their relatively complicated triggering constraints. Covering algorithms are a class of inductive learning techniques that successively generalize input examples until the generalization produces errors. As a result, CRYSTAL begins by generating the most specific concept node possible for every phrase to be extracted in the training texts. It then progresses through the concept nodes one by one. For each concept node, C, CRYSTAL finds the most similar concept node, C', and relaxes the constraints of each just enough to unify C and C'. The new extraction pattern, P, is tested against the training corpus. If its error rate is less than some prespecified threshold, P is added to the set, replacing C and C'. The process is repeated on P until the error tolerance is exceeded. At that point, CRYSTAL moves on to the next pattern in the original set. CRYSTAL was initially used to derive extraction patterns for a medical diagnosis domain where it achieved precision levels ranging from 50–80% and recall levels ranging from 45–75% depending on how the error tolerance threshold was set.

Although AutoSlog, PALKA, and CRYSTAL learn extraction patterns in the form of semantic case frames, each uses a different learning strategy. AutoSlog creates extraction patterns by specializing a small set of general linguistic patterns; CRYSTAL generalizes complex, but maximally specific linguistic contexts; and PALKA performs both generalization and specialization of an initial extraction pattern. Where AutoSlog makes no attempt to limit the number of extraction patterns created, CRYSTAL's covering algorithm derives the minimum number of patterns that cover the examples in the training corpus. In addition, CRYSTAL and PALKA employ automated feedback for the learning algorithm; AutoSlog requires human perusal of proposed patterns. CRYSTAL and PALKA, on the other hand, require more background knowledge in the form of a possibly domain-specific semantic class hierarchy, a lexicon that indicates semantic class information for each word, and, in the case of PALKA, a set of trigger words. The parsers of both systems must also be able to accurately assign semantic class information to words in an incoming text. While AutoSlog's patterns perform best when semantic class information is available, the learning algorithm and the resulting concept nodes can still operate effectively when no semantic class information can be obtained.

There have been a few additional attempts to learn extraction patterns. Huffman's LIEP system [1996] learns patterns that recognize semantic relationships between two target noun phrases, i.e. between two slot fillers of an information extraction output template. The patterns describe the syntactic context that falls between the target noun phrases as well as the semantic class of the heads of the target phrases and all intervening phrases. Cardie [1993] used standard symbolic machine learning algorithms (decision tree induction and a k-nearest neighbor algorithm) to identify the trigger word for an extraction pattern, the general linguistic context in which the pattern would be applied, and the type of concept that the pattern would identify. Califf and Mooney [1997] have recently applied relational learning techniques to acquire extraction patterns from newsgroup articles that describe job postings. Like CRYSTAL, their RAPIER system operates by generalizing an initial set of specific patterns. Unlike any of the previously mentioned systems, however, RAPIER learns patterns that specify constraints at the word level rather than the constituent level. As a result, only a part-of-speech tagger is required to process input texts.

While much progress has been made on learning extraction patterns, there are still many research issues to be resolved. Existing methods work well when the information to be extracted is explicitly denoted as a string in the text, but major extensions would be required to handle "set fills" — slot fillers that are chosen from a predefined list rather than occuring directly in the text. Furthermore, existing methods focus on the extraction of noun phrases. It is not clear that the same methods would work well for domains in which the extracted information is another syntactic type or is a component of a constituent rather than a complete constituent (e.g., a group of noun modifiers in a noun phrase). Finally, few of the methods described above have been evaluated on the same

information extraction tasks under the same conditions. Until a direct comparison of techniques is available, it will remain difficult to determine the relative advantages of one technique over another. A related open problem in the area is to determine, *a priori*, which method for learning extraction patterns will give the best results in a new extraction domain.

4 Coreference Resolution and Template Generation

In comparison to empirical methods for learning extraction patterns, substantially less research has tackled the problems of Merging/Coreference Resolution and Template Generation. As mentioned earlier, the goal of the coreference component is to determine when two phrases refer to the same entity. Although this may not appear to be a difficult task, consider the following text from the MUC-6 corporate management succession domain (MUC-6, 1995). In this text all of the bracketed segments are coreferential:

> [Motor Vehicles International Corp.] announced a major management shake-up ... [MVI] said the chief executive officer has resigned ... [The Big 10 auto maker] is attempting to regain market share. ... [It] will announce significant losses for the fourth quarter ... A [company] spokesman said [they] are moving [their] operations to Mexico in a cost-saving effort. ... [MVI, [the first company to announce such a move since the passage of the new international trade agreement],] is facing increasing demands from unionized workers. ... [Motor Vehicles International] is [the biggest American auto exporter to Latin America].

The passage shows the wide range of linguistic phenomena that comprise coreference resolution including proper names, aliases, definite noun phrases, definite descriptions, pronouns, predicate nominals, and appositives. Unfortunately, different factors may play a role in handling each type of reference. In fact, discourse processing, and coreference in particular, has been cited as a major weakness of existing information extraction systems. One problem is that most systems use manually generated heuristics to determine when two phrases describe the same entity, but generating good heuristics that cover all types of reference resolution is challenging. In particular, few discourse theories have been evaluated empirically, and, as a result, it is not clear what information to include in the heuristics. It is also difficult to design heuristics that combine multiple coreference cues effectively given that the relative importance of each piece of information is unknown. Furthermore, most computational approaches to coreference resolution assume as input fully parsed sentences, often with additional linguistic attributes like grammatical function and thematic role information. Information extraction systems do not normally have such detailed parse information available: the robust partial parsing algorithms employed by most information extraction systems offer wider coverage in exchange for less syntactic information. A further complication in developing trainable coreference components for an information extraction system is that discourse analysis is based on information discerned by earlier phases of processing. This means that any coreference algorithms must take into account the accumulated errors of the earlier phases as well as the fact that some information that would aid the coreference task may be missing. Finally, the coreference component of an information extraction system must be able to handle the myriad forms of coreference across different domains.

Empirical methods for coreference were designed to address these problems. Unlike the methods for learning extraction patterns, algorithms for building automatically trainable coreference resolution systems have not required the development of learning algorithms designed specifically for the task. By recasting the coreference problem as a classification task, any of a number of standard inductive learning algorithms can be employed. Given two phrases and the context in which they occur, for example, the coreference learning algorithm must classify the phrases with respect to whether or not they refer to the same object. Below, we describe two systems that use inductive classification techniques to automatically acquire coreference resolution heuristics: MLR (Aone and Bennett, 1995) and RESOLVE (McCarthy and Lehnert, 1995).

Both MLR (Machine Learning-based Resolver) and RESOLVE use the same general approach, which is depicted in Figure 5. First, a training corpus is annotated with coreference information.

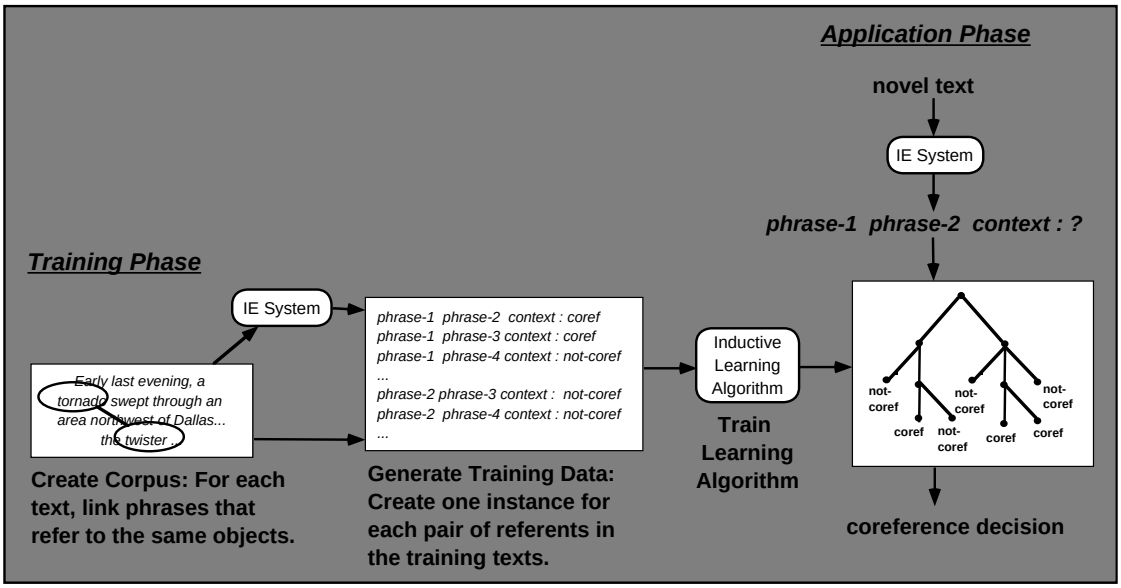

Figure 5: A Machine Learning Approach to Coreference Resolution.

Namely, *all* phrases that refer to the same object are to be linked via the annotations. Alternatively, just the "best" (usually the most recent) antecedent for each referent is marked. Training examples for presentation to the machine learning algorithm are then created from the corpus. There will be one instance for every possible pairing of referents in the training texts: some of these are "positive" examples in that they correspond to phrases that *are* coreferent; others are "negative" examples in that they correspond to phrases that are *not* referring to the same object. The exact form of the instances depends on the learning algorithm, but for the inductive learning algorithm used by MLR and RESOLVE, the training examples contain: (1) a list of features, or attribute-value pairs, that describe the phrases under consideration and the context in which they occur, and (2) supervisory information in the form of a "class" value that indicates whether or not the two phrases are coreferent. The specific features used depend on the kinds of information available to the information extraction system when the coreference decision must be made. More details on the creation of training data will be included below.

Once the data set has been derived from the corpus, it is presented to the machine learning algorithm, which uses the examples to derive a concept description for the coreference resolution task. Figure 5 shows a concept description in the form of a decision tree, but the actual form depends on the particular learning algorithm employed. The idea is that, after training, this concept description can be used to decide whether two phrases in an *unseen* text refer to the same object. This process is shown as the Application Phase in Figure 5. The information extraction system processes a new text and reaches a point where coreference decisions must be made. For each such decision, the NLP system creates a "test instance." The test instance employs the same feature set as the training instances: its features describe a discourse entity, its possible antecedent, and their shared context. The test instance is given to the learned concept description for classification as either coreferent or not; and the decision is returned to the information extraction system.

Both MLR and RESOLVE use this general method for automatically constructing coreference components for their information extraction systems. Both use the widely available C4.5 decision tree induction system (Quinlan, 1992) as the inductive learning component. There are, however, a number of differences in how each system instantiated and evaluated the general approach of Figure 5. McCarthy tested RESOLVE on the MUC-5 business joint ventures corpus (English version); Aone and Bennett tested MLR on the Japanese corpus for the same information extraction domain. The evaluation of MLR focused on anaphors involving entities tagged as organizations (e.g., companies, governments) by the Sentence Analysis phase of their information extraction

system. The evaluation of RESOLVE focused on entities tagged as JV-ENTITIES — entities that were organizations and that had been identified as a party in a joint venture by the Extraction component.

Both systems used feature representations that relied only on information that earlier phases of analysis could provide. However, MLR's data set was generated automatically by its information extraction system, while RESOLVE was evaluated using a manually generated, noise-free data set. In addition, the feature sets of each system varied markedly. MLR's training and test instances were described in terms of 66 features that describe: (1) lexical features of each phrase (e.g., whether one phrase contains a character subsequence of the other), (2) the grammatical role of the phrases, (3) semantic class information, and (4) relative positional information. While all attributes of the MLR representation are domain-independent, the values for some attributes may be domain-specific. RESOLVE's instance representation, on the other hand, contains a number of features that are unabashedly domain-specific. Its representation includes eight features including: whether or not each phrase contains a proper name (2 features), whether one or both phrases refer to the entity formed by a joint venture (3 features), whether one phrase contains an alias of the other (1 feature), whether the phrases have the same base noun phrase (1 feature), and whether the phrases originate from the same sentence (1 feature). Note that a number of RESOLVE's features correspond to those used in MLR, e.g., the Alias feature of RESOLVE vs. the Character Subsequence feature of MLR.

RESOLVE and MLR were evaluated using data sets derived from 50 and 250 texts, respectively. RESOLVE achieved recall and precision levels of 80-85% and 87-92% (depending on whether the decision tree was pruned or not). A baseline system that always assumed that the candidate phrases were **not** coreferent would also achieve relatively high scores given that negative examples comprised 74% of the RESOLVE data set. MLR, on the other hand, achieved recall and precision levels of 67-70% and 83-88% (depending on the parameter settings of the training configuration). For both MLR and RESOLVE, recall and precision are measured with respect to the coreference task only, not the full information extraction task.

Without additional experiments, it is impossible to know whether the differences in results depend on the language (English vs. Japanese), the slight variations in training/testing methodology, the degree of noise in the data, or the feature sets employed. Interestingly, MLR does well because a single feature — the Character Subsequence feature — can reliably predict coreference for phrases that are proper names for organizations and these comprise almost half of the instances in the data set. Performance on non-proper-name organization referents was much lower. Definite noun phrases, for example, reached only 44% recall and 60% precision. Nevertheless, an important result for both MLR and RESOLVE was that each significantly outperformed a coreference system that had been developed manually for their information extraction systems.

In a subsequent evaluation, the RESOLVE system competed in the MUC-6 coreference competition where it achieved scores of 41–44% recall and 51–59% precision after training on only 25 texts. This was somewhat below the five best systems, which achieved 51–63% recall and 62-72% precision. All of the better-performing systems, however, employed manually encoded coreference algorithms. Like some of the manually coded systems, RESOLVE only attempted to resolve references to people and organizations. In fact, it was estimated that a good proper name/alias recognizer would have produced a coreference systems with relatively good performance — about 30% recall and possibly 90% precision. One should note, however, that the interannotator agreement for marking coreference in 17 articles was found to be 80% recall and 82% precision, with definite descriptions (e.g., ... [MVI, [the first company to announce such a move since the passage of the new international trade agreement]]) and bare nominals (e.g., "A [company] spokesman") accounting for most of the discrepancies.

Overall, the results for coreference resolution are promising. They show that it is possible to develop automatically trainable coreference systems that can compete favorably with manually designed systems. In addition, they show that specially designed learning algorithms need not be developed because standard inductive machine learning algorithms may be up to the challenge. There is an additional advantage to applying symbolic machine learning techniques to problems in natural language understanding: they offer a mechanism for evaluating the usefulness of different knowledge sources for any task in an NLP system that can be described as a classification

problem. Examination of the coreference decision trees created by C4.5, for example, will indicate which knowledge sources are more important for the task: the knowledge source corresponding to a feature tested at node i in the tree is probably more important than the knowledge sources corresponding to the features tested below it in the tree. Furthermore, once the data set is created, it is a simple task to run multiple variations of the learning algorithm, giving each variation access to a different subset of features. As a result, empirical methods offer data-driven feedback for linguistic theories and system developers alike.

Still, much research remains to be done. The machine learning approach to coreference should be tested on additional types of anaphors using a variety of feature sets, including feature sets that require no domain-specific information. In addition, if the approach is to offer a general, task-independent solution to the coreference problem, then the role of domain-specific information for coreference resolution must be empirically determined and the methods must be evaluated outside the context of information extraction. The relative effect of errors from the preceding phases of text analysis on learning algorithm performance must also be investigated.

There have been few attempts to use empirical methods for other discourse-level problems that arise in information extraction. BBN has developed a probabilistic method for determining paragraph relevance for its information extraction system (Weischedel *et al.*, 1993); they then use the device to control the recall/precision tradeoff. Cardie has used symbolic machine learning techniques to learn relative pronoun disambiguation heuristics (Cardie, 1992b; Cardie, 1992a). This allows the information extraction system to process a sentence like "Castellar was kidnapped by **members of the ELN**, *who* attacked the mayor in his office." and infer that "members of the ELN" is the actor of the kidnapping as well as the implicit actor of the attack in the second clause. Two trainable systems that simultaneously tackle Merging and Template Generation have also been developed: TTG (Dolan *et al.*, 1991) and Wrap-Up (Soderland and Lehnert, 1994). Both systems generate a series of decision trees, each of which handles some piece of the template generation or merging tasks, e.g., deciding whether to merge two templates into one, or deciding when to split an existing template into two or more templates. Wrap-Up used 91 decision trees to make these decisions for the MUC-5 microelectronics domain based on features of the entities extracted from each clause in an input text. One decision tree, for example, decides whether a Lithography template and an Equipment object should be linked in the final collection of output templates. Unfortunately, the information extraction systems that used these trainable discourse components did not perform nearly as well as systems that used manually generated Merging and Template Generation subsystems. Additional research is needed to determine the feasibility of an entirely trainable discourse component. Finally, statistical approaches to template merging are also beginning to surface. Kehler [1997], for example, introduced a method for assigning a probability distribution to coreference relationships as encoded in competing sets of output templates. His initial experiments indicate that the method compares favorably with the greedy approach to template merging that is used in SRI's FASTUS information extraction system.

5 Future Directions

Research in information extraction is very new. Research in applying learning algorithms to problems in information extraction is even newer: We are only beginning to understand the techniques for automatically acquiring both domain-independent and domain-dependent knowledge for these task-driven systems. As a result, there are a number of exciting directions that the field may take. First, like the trends in statistical language learning, a next step would be to explore unsupervised learning algorithms as a means for sidestepping the lack of large, annotated corpora for information extraction tasks. In general, there is a dirth of learning algorithms that deal effectively with the relatively small amounts of data available to developers of information extraction systems. A related, but slightly different direction of research is to focus on developing techniques that allow end-users to quickly train information extraction systems for their own needs through interaction with the system over time, completely eliminating the need for intervention by NLP system developers. Many new learning methods will be needed to succeed in this task, not the least of which

are techniques that make direct use of the answer keys of an information extraction training corpus to automatically tune *all* components of the extraction system for a new domain. The demand for information extraction systems in industry, government, education, and for personal use is spiraling as more and more text becomes available on-line. The challenge for empirical methods in NLP is to continue to match that demand by developing additional natural language learning techniques that replace manual coding efforts with automatically trainable components and that make it increasingly faster and easier to build accurate and robust information extraction systems in new domains.

6 Acknowledgments

Preparation of this article was supported in part by NSF CAREER Award IRI–9624639.

References

(Aone and Bennett, 1995) Aone, Chinatsu and Bennett, William 1995. Evaluating Automated and Manual Acquisition of Anaphora Resolution Strategies. In *Proceedings of the 33rd Annual Meeting of the ACL*. Association for Computational Linguistics. 122–129.

(Brill, 1995) Brill, Eric 1995. Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging. *Computational Linguistics* 21(4):543–565.

(Califf and Mooney, 1997) Califf, M. E. and Mooney, R. J. 1997. Relational Learning of Pattern-Match Rules for Information Extraction. In *Proceedings of the ACL Workshop on Natural Language Learning*, Madrid, Spain. 9–15.

(Cardie and Lehnert, 1991) Cardie, C. and Lehnert, W. 1991. A Cognitively Plausible Approach to Understanding Complicated Syntax. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, Anaheim, CA. AAAI Press / MIT Press. 117–124.

(Cardie, 1992a) Cardie, C. 1992a. Corpus-Based Acquisition of Relative Pronoun Disambiguation Heuristics. In *Proceedings of the 30th Annual Meeting of the ACL*, University of Delaware, Newark, DE. Association for Computational Linguistics. 216–223.

(Cardie, 1992b) Cardie, C. 1992b. Learning to Disambiguate Relative Pronouns. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, San Jose, CA. AAAI Press / MIT Press. 38–43.

(Cardie, 1993) 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*, Washington, DC. AAAI Press / MIT Press. 798–803.

(Charniak, 1993) Charniak, Eugene 1993. *Statistical Language Learning*. MIT Press, Cambridge, MA.

(Chinchor *et al.*, 1993) Chinchor, N.; Hirschman, L.; and Lewis, D. 1993. Evaluating Message Understanding Systems: An Analysis of the Third Message Undestanding Conference (MUC-3). *Computational Linguistics* 19(3):409–449.

(Craven *et al.*, 1997) Craven, M.; Freitag, D.; McCallum, A.; Mitchell, T.; Nigam, K.; and Quek, C.Y. 1997. Learning to extract symbolic knowledge from the world wide web. Internal report, School of Computer Science, CMU.

(Daelemans *et al.*, 1996) Daelemans, W.; Zavrel, J.; P., Berck; and S., Gillis 1996. MBT: A Memory-Based Part of Speech Tagger Generator. In Ejerhed, Eva and Dagan, Ido, , editor, *Proceedings of the Fourth Workshop on Very Large Corpora*. ACL SIGDAT. 14–27.

(Dolan *et al.*, 1991) Dolan, C.; Goldman, S.; Cuda, T.; and Nakamura, A. 1991. Hughes Trainable Text Skimmer: Description of the TTS System as Used for MUC-3. In *Proceedings of the Third Message Understanding Conference (MUC-3)*, San Mateo, CA. Morgan Kaufmann. 155–162.

(Glasgow *et al.*, 1997) Glasgow, B.; Mandell, A.; Binney, D.; Ghemri, L.; and Fisher, D. 1997. MITA: An Information Extraction Approach to Analysis of Free-form Text in Life Insurance Applications. In *Ninth Conference on Innovative Applications of Artificial Intelligence*. AAAI Press / MIT Press. 992–999.

(Holowczak and Adam, 1997) Holowczak, R. D. and Adam, N. R. 1997. Information Extraction based Multiple-Category Document Classification for the Global Legal Information Network. In *Ninth Conference on Innovative Applications of Artificial Intelligence*. AAAI Press / MIT Press. 1013–1018.

(Huffman, 1996) Huffman, Scott 1996. Learning Information Extraction Patterns from Examples. In Wermter, Stefan; Riloff, Ellen; and Scheler, Gabriele, editors, *Symbolic, connectionist, and statistical approaches to learning for natural language processing*, Lecture Notes in Artificial Intelligence Series. Springer. 246–260.

(Kehler, 1997) Kehler, Andrew 1997. Probabilistic Coreference in Information Extraction. In Cardie, Claire and Weischedel, Ralph, editors, *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. 163–173.

(Kim and Moldovan, 1995) Kim, Jun-Tae and Moldovan, Dan I. 1995. Acquisition of linguistic patterns for knowledge-based information extraction. *IEEE Transactions on Knowledge and Data Engineering* 7(5):713–724.

(Lehnert and Sundheim, 1991) Lehnert, W. and Sundheim, B. 1991. A performance evaluation of text analysis technologies. *AI Magazine* 12(3):81–94.

(Lehnert *et al.*, 1991) Lehnert, W.; Cardie, C.; Fisher, D.; Riloff, E.; and Williams, R. 1991. University of Massachusetts: Description of the CIRCUS System as Used in MUC-3. In *Proceedings of the Third Message Understanding Conference (MUC-3)*, San Mateo, CA. Morgan Kaufmann. 223–233.

(Lehnert *et al.*, 1992) Lehnert, W.; Cardie, C.; Fisher, D.; McCarthy, J.; Riloff, E.; and Soderland, S. 1992. University of Massachusetts: Description of the CIRCUS System as Used in MUC-4. In *Proceedings of the Fourth Message Understanding Conference (MUC-4)*, San Mateo, CA. Morgan Kaufmann. 282–288.

(Lehnert, 1990) Lehnert, W. 1990. Symbolic/Subsymbolic Sentence Analysis: Exploiting the Best of Two Worlds. In Barnden, J. and Pollack, J., editors, *Advances in Connectionist and Neural Computation Theory*. Ablex Publishers, Norwood, NJ. 135–164.

(Magerman, 1995) Magerman, David M. 1995. Statistical Decision-Tree Models for Parsing. In *Proceedings of the 33rd Annual Meeting of the ACL*. Association for Computational Linguistics. 276–283.

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

(McCarthy and Lehnert, 1995) McCarthy, Joseph F. and Lehnert, Wendy G. 1995. Using Decision Trees for Coreference Resolution. In Mellish, C., editor, *Proceedings of the Fourteenth International Conference on Artificial Intelligence*. 1050–1055.

(MUC-3, 1991) *Proceedings of the Third Message Understanding Conference (MUC-3)*. Morgan Kaufmann, San Mateo, CA.

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

(MUC-5, 1994) *Proceedings of the Fifth Message Understanding Conference (MUC-5)*. Morgan Kaufmann, San Mateo, CA.

(MUC-6, 1995) *Proceedings of the Sixth Message Understanding Conference (MUC-6)*. Morgan Kaufmann, San Francisco, CA.

(Quinlan, 1992) Quinlan, J. R. 1992. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

(Ramshaw and Marcus, 1995) Ramshaw, Lance A. and Marcus, Mitchell P. 1995. Text Chunking using Transformation-Based Learning. In *Proceedings of the 33rd Annual Meeting of the ACL*. Association for Computational Linguistics. 82–94.

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

(Riloff, 1996) Riloff, E. 1996. Automatically Generating Extraction Patterns from Untagged Text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Portland, OR. AAAI Press / MIT Press. 1044–1049.

(Soderland and Lehnert, 1994) Soderland, S. and Lehnert, W. 1994. Corpus-Driven Knowledge Acquisition for Discourse Analysis. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Seattle, WA. AAAI Press / MIT Press. 827–832.

(Soderland *et al.*, 1995) Soderland, S.; Fisher, D.; Aseltine, J.; and Lehnert, W. 1995. CRYSTAL: Inducing a Conceptual Dictionary. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence*. AAAI Press / MIT Press. 1314–1319.

(Soderland *et al.*, 1995) Soderland, S.; Aronow, D.; Fisher, D.; Aseltine, J.; and Lehnert, W. 1995. Machine learning of text analysis rules for clinical records. Technical Report TE-39, University of Massachusetts.

(Soderland, 1997) Soderland, S. 1997. Learning to extract text-based information from the world wide web. In *Proceedings of Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*. AAAI Press. 251–254.

(Thompson *et al.*, 1997) Thompson, C. A.; Mooney, R. J.; and Tang, L. R. 1997. Learning to parse natural language database queries into logical form. In *Proceedings of the ML-97 Workshop on Automata Induction, Grammatical Inference, and Language Acquisition*. ACL.

(Weischedel *et al.*, 1993) Weischedel, R.; Ayuso, D.; Boisen, S.; Fox, H.; Matsukawa, T.; Papageorgiou, C.; MacLaughlin, D.; Sakai, T.; Abe, H. J. Hosihi; Miyamoto, Y.; and Miller, S. 1993. BBN's PLUM Probabilistic Language Understanding System. In *Proceedings, TIPSTER Text Program (Phase I)*, San Mateo, CA. Morgan Kaufmann. 195–208.

(Will, 1993) Will, Craig A. 1993. Comparing Human and Machine Performance for Natural Language Information Extraction: Results from the TIPSTER Text Evaluation. In *Proceedings, TIPSTER Text Program (Phase I)*, San Mateo, CA. Morgan Kaufmann. 179–194.

(Zelle and Mooney, 1994) Zelle, J. and Mooney, R. 1994. Inducing Deterministic Prolog Parsers from Treebanks: A Machine Learning Approach. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Seattle, WA. AAAI Press / MIT Press. 748–753.