Unsupervised Discovery of Scenario-Level Patterns for Information Extraction

Roman Yangarber
roman@cs.nyu.edu
Courant Institute of Mathematical Sciences
New York University

Ralph Grishman
grishman@cs.nyu.edu
Courant Institute of Mathematical Sciences
New York University

Pasi Tapanainen †
tapanain@conexor.fi
† Conexor Oy
Helsinki, Finland

Silja Huttunen ‡
sihuttun@ling.helsinki.fi
‡ University of Helsinki
Finland

Abstract

Information Extraction (IE) systems are commonly based on pattern matching. Adapting an IE system to a new scenario entails the construction of a new pattern base—a time-consuming and expensive process. We have implemented a system for finding patterns automatically from *un-annotated* text. Starting with a small initial set of *seed* patterns proposed by the user, the system applies an incremental discovery procedure to identify new patterns. We present experiments with evaluations which show that the resulting patterns exhibit high precision and recall.

0 Introduction

The task of Information Extraction (IE) is the selective extraction of meaning from free natural language text.[1] "Meaning" is understood here in terms of a fixed set of semantic objects—*entities*, *relationships* among entities, and *events* in which entities participate. The semantic objects belong to a small number of *types*, all having fixed regular structure, within a fixed and closely circumscribed subject domain. The extracted objects are then stored in a relational database. In this paper, we use the nomenclature accepted in current IE literature; the term *subject domain* denotes a class of textual documents to be processed, e.g., "business news," and *scenario* denotes the specific topic of interest within the domain, i.e., the set of facts to be extracted. One example of a scenario is "management succession," the topic of MUC-6 (the Sixth Message Understanding Conference); in this scenario the system seeks to identify events in which corporate managers left their posts or assumed new ones. We will consider this scenario in detail in a later section describing experiments.

IE systems today are commonly based on pattern matching. The patterns are regular expressions, stored in a "pattern base" containing a general-purpose component and a substantial domain- and scenario-specific component.

Portability and performance are two major problem areas which are recognized as impeding widespread use of IE. This paper presents a novel approach, which addresses *both* of these problems by automatically discovering good patterns for a new scenario. The viability of our approach is tested and evaluated with an actual IE system.

In the next section we describe the problem in more detail in the context of our IE system; sections 2 and 3 describe our algorithm for pattern discovery; section 4 describes our experimental results, followed by comparison with prior work and discussion, in section 5.

1 The IE System

Our IE system, among others, contains a a back-end core engine, at the heart of which is a regular-expression pattern matcher. The engine draws on attendant knowledge bases (KBs) of varying degrees of domain-specificity. The KB components are commonly factored out to make the systems portable to new scenarios. There are four customizable knowledge bases in our IE system: the *Lexicon* contains general dictionaries and scenario-specific terms; the *concept base* groups terms into classes; the *predicate base* describes the logical structure of events to be extracted, and the *pattern base* contains patterns that catch the events in text.

Each KB has a substantial domain-specific component, which must be modified when mov-

[1] For general references on IE, cf., e.g., (Pazienza, 1997; muc, 1995; muc, 1993).

ing to new domains and scenarios. The system allows the *user* (i.e. scenario developer) to start with example sentences in text which contain events of interest, the *candidates*, and generalize them into patterns. However, the user is ultimately responsible for *finding* all the candidates, which amounts to manually processing example sentences in a very large training corpus. Should s/he fail to provide an example of a particular class of syntactic/semantic construction, the system has no hope of recovering the corresponding events. Our experience has shown that (1) the process of discovering candidates is highly expensive, and (2) gaps in patterns directly translate into gaps in coverage.

How can the system help automate the process of *discovering* new good candidates? The system should find examples of all common linguistic constructs relevant to a scenario. While there has been prior research on identifying the primary lexical patterns of a sub-language or corpus (Grishman et al., 1986; Riloff, 1996), the task here is more complex, since we are typically not provided in advance with a sub-corpus of relevant passages; these passages must themselves be found as part of the discovery process. The difficulty is that one of the best indications of the relevance of the passages is precisely the presence of these constructs. Because of this circularity, we propose to acquire the constructs and passages *in tandem*.

2 Solution

We outline our procedure for automatic acquisition of patterns; details are elaborated in later sections. The procedure is *unsupervised* in that it does not require the training corpus to be manually annotated with events of interest, nor a *pre-classified corpus* with relevance judgements, nor any feedback or intervention from the user[2]. The idea is to combine IR-style document selection with an *iterative relaxation* process; this is similar to techniques used elsewhere in NLP, and is inspired in large part, if remotely, by the work of (Kay and Röscheisen, 1993) on automatic alignment of sentences and words in a bilingual corpus. There, the reasoning was: sentences that are translations of each other are good indicators that words they contain are translation pairs; conversely, words that are translation pairs indicate that the sentences which contain them correspond to one another.

In our context, we observe that documents that are relevant to the scenario will necessarily contain good patterns; conversely, good patterns are strong indicators of relevant documents. The outline of our approach is as follows.

0. Given: (1) a large corpus of *un-annotated and un-classified* documents in the domain; (2) an initial set of trusted scenario patterns, as chosen *ad hoc* by the user—the *seed*; as will be seen, the seed can be quite small—two or three patterns seem to suffice. (3) an initial (possibly empty) set of concept classes

1. The pattern set induces a binary partition (a *split*) on the corpus: on any document, either zero or more than zero patterns will match. Thus the universe of documents, U, is partitioned into the relevant sub-corpus, R, vs. the non-relevant sub-corpus, $\overline{R} = U - R$, with respect to the given pattern set. Actually, the documents are assigned weights which are 1 for documents matched by the trusted seed, and 0 otherwise.[3]

2. Search for new candidate patterns:

   (a) Automatically convert each sentence in the corpus into a set of candidate patterns.[4]

   (b) Generalize each pattern by replacing each lexical item which is a member of a concept class by the class name.

   (c) Working from the relevant documents, select those patterns whose distribution is strongly correlated with other relevant documents (i.e., much more

[2] however, it may be supervised after each iteration, where the user can answer yes/no questions to improve the quality of the results

[3] R represents the trusted truth through the discovery iterations, since it was induced by the manually-selected seed.

[4] Here, for each clause in the sentence we extract a tuple of its major roles: the head of the subject, the verb group, the object, object complement, as described below. This tuple is considered to be a pattern for the present purposes of discovery; it is a skeleton for the rich, syntactically transformed patterns our system uses in the extraction phase.

densely distributed among the relevant documents than among the non-relevant ones). The idea is to consider those candidate patterns, p, which meet the *density* criterion:

$$\frac{|H \cap R|}{|H \cap U|} \gg \frac{|R|}{|U|}$$

where $H = H(p)$ is the set of documents where p hits.

(d) Based on co-occurrence with the chosen patterns, extend the concept classes.

3. *Optional:* Present the new candidates and classes to the user for review, retaining those relevant to the scenario.

4. The new pattern set induces a new partition on the corpus. With this pattern set, return to step 1. Repeat the procedure until no more patterns can be added.

3 Methodology

3.1 Pre-processing: Normalization

Before applying the discovery procedure, we subject the corpus to several stages of preprocessing. First, we apply a name recognition module, and replace each name with a token describing its class, e.g. *C-Person*, *C-Company*, etc. We collapse together all numeric expressions, currency values, dates, etc., using a single token to designate each of these classes.

3.2 Syntactic Analysis

We then apply a parser to perform syntactic normalization to transform each clause into a common predicate-argument structure. We use the general-purpose dependency parser of English, based on the FDG formalism (Tapanainen and Järvinen, 1997) and developed by the Research Unit for Multilingual Language Technology at the University of Helsinki, and Conexor Oy. The parser (modified to understand the name labels attached in the previous step) is used for reducing such variants as passive and relative clauses to a tuple, consisting of several elements.

1. For each claus, the first element is the subject, a "semantic" subject of a non-finite sentence or agent of the passive.[5]

2. The second element is the verb.

3. The third element is the object, certain object-like adverbs, subject of the passive or subject complement[6]

4. The fourth element is a phrase which refers to the object or the subject. A typical example of such an argument is an object complement, such as *Company named John Smith* **president**. Another instance is the so-called *copredicative* (Nichols, 1978), in the parsing system (Järvinen and Tapanainen, 1997). A copredicative refers to a subject or an object, though this distinction is typically difficult to resolve automatically.[7]

Clausal tuples also contain a locative modifier, and a temporal modifier. We used a corpus of 5,963 articles from the Wall Street Journal, randomly chosen. The parsed articles yielded a total of 250,000 clausal tuples, of which 135,000 were distinct.

3.3 Generalization and Concept Classes

Because tuples may not repeat with sufficient frequency to obtain reliable statistics, each tuple is reduced to a set of pairs: e.g., a verb-object pair, a subject-object pair, etc. Each pair is used as a generalized pattern during the candidate selection stage. Once we have identified pairs which are relevant to the scenario, we use them to construct or augment concept classes, by grouping together the missing roles, (for example, a class of verbs which occur with a relevant subject-object pair: "*company* {hire/fire/expel...} *person*"). This is similar to work by several other groups which aims to induce semantic classes through syntactic co-occurrence analysis (Riloff and Jones, 1999; Pereira et al., 1993; Dagan et al., 1993; Hirschman et al., 1975), although in our case the contexts are limited to selected patterns, relevant to the scenario.

[5] E.g., "**John** sleeps", "John is appointed by **Company**", "I saw a **dog** which sleeps", "She asked **John** to buy a car".

[6] E.g., "**John** is appointed by Company", "John is the **president** of Company", "I saw a **dog** which sleeps", The **dog** which I saw sleeps.

[7] For example, "She gave us our coffee **black**", "Company appointed John Smith as **president**".

3.4 Pattern Discovery

Here we present the results from experiments we conducted on the MUC-6 scenario, "management succession". The discovery procedure was seeded with a small pattern set, namely:

Subject	*Verb*	*Direct Object*
C-Company	C-Appoint	C-Person
C-Person	C-Resign	—

Here *C-Company* and *C-Person* denote semantic classes containing named entities of the corresponding semantic types. *C-Appoint* denotes a class of verbs, containing four verbs { *appoint, elect, promote, name*}; *C-Resign* = { *resign, depart, quit, step-down* }.

During a single iteration, we compute the score[8], $L(p)$, for each candidate pattern p:

$$L(p) = P_c(p) \cdot \log |H \cap R| \qquad (1)$$

where R denotes the relevant subset, and $H = H(p)$ the documents matching p, as above, and $P_c(p) = \frac{|H \cap R|}{|H|}$ is the conditional probability of relevance. We further impose two *support* criteria: we distrust such frequent patterns where $|H \cap U| > \alpha|U|$ as uninformative, and rare patterns for which $|H \cap R| < \beta$ as noise.[9] At the end of each iteration, the system selects the pattern with the highest score, $L(p)$, and adds it to the seed set. The documents which the winning pattern hits are added to the relevant set. The pattern search is then restarted.

3.5 Re-computation of Document Relevance

The above is a simplification of the actual procedure, in several important respects.

Only generalized patterns are considered for candidacy, with one or more slots filled with wild-cards. In computing the score of the generalized pattern, we do not take into consideration *all* possible values of the wild-card role. We instead constrain the wild-card to those values which themselves in turn produce patterns with high scores. These values then become members of a new class, which is output in tandem with the winning pattern[10]

[8] similarly to (Riloff, 1996)

[9] U denotes the universe of documents. We used $\alpha = 0.1$ and $\beta = 2$.

[10] The classes are currently unused by subsequent iterations; this important issue is considered in future work.

Documents are assigned relevance scores on a scale between 0 and 1. The seed patterns are accepted as ground truth; thus the documents they match have relevance 1. On subsequent iterations, the newly accepted patterns are not trusted as absolutely. On iteration number $i+1$, each pattern p is assigned a precision measure, based on the relevance of the documents it matches:

$$Prec^{i+1}(p) = \frac{1}{|H(p)|} \cdot \sum_{d \in H(p)} Rel^i(d) \qquad (2)$$

where $Rel^i(d)$ is the relevance of the document from the previous iteration, and $H(p)$ is the set of documents where p matched. More generally, if K is a classifier consisting of a *set* of patterns, we can define $H(K)$ as the set of documents where *all* of patterns $p \in K$ match, and the "cumulative" precision[11] of K as

$$Prec^{i+1}(K) = \frac{1}{|H(K)|} \cdot \sum_{d \in H(K)} Rel^i(d) \qquad (3)$$

Once the new winning pattern is accepted, the relevance scores of the documents are readjusted as follows. For each document d which is matched by some (non-empty) subset of the currently accepted patterns, we can view that subset of patterns as a classifier $K_d = \{p_j\}$. These patterns determine the new relevance score of the document as

$$Rel^{i+1}(d) = \max\left(Rel^i(d), Prec^{i+1}(K_d)\right) \qquad (4)$$

This ensures that the relevance score grows monotonically, and only when there is sufficient positive evidence, as the patterns in effect vote "conjunctively" on the documents. The results which follow use this measure.

Thus in the formulas above, R is not simply the count of the relevant documents, but is rather their *cumulative relevance*. The two formulas, (3) and (4), capture the mutual dependency of patterns and documents; this recomputation and growing of precision and relevance scores is at the heart of the procedure.

[11] Of course, this measure is defined only when $H(K) \neq \emptyset$.

4 Results

An objective measure of goodness of a pattern is not trivial to establish since the patterns cannot be used for extraction directly, without being properly incorporated into the knowledge base. Thus, the discovery procedure does not lend itself easily to MUC-style evaluations, since a pattern lacks information about which events it induces and which slots its arguments should fill.

However, it is possible to apply some objective measures of performance. One way we evaluated the system is by noting that in addition to growing the pattern set, the procedure also grows the relevance of documents. The latter *can* be objectively evaluated.

We used a test corpus of 100 MUC-6 formal-training documents (which were included in the main development corpus of about 6000 documents) plus another 150 documents picked at random from the main corpus and judged by hand. These judgements constituted the ground truth and were used only for evaluation, (not in the discovery procedure).

4.1 Text Filtering

Figure 1 shows the recall/precision measures with respect to the test corpus of 250 documents, over a span of 60 generations, starting with the seed set in table 3.4. The seed patterns matched 184 of the 5963 documents, yielding an initial recall of .11 and precision of .93; by the last generation it searched through 982 documents with non-zero relevance, and ended with .80 precision and .78 recall. This facet of the discovery procedure is closely related to the MUC "text-filtering" sub-task, where the systems are judged at the level of *documents* rather than event slots. It is interesting to compare the results with other MUC-6 participants, shown anonymously in figure 2. Considering recall and precision separately, the discovery procedure attains values comparable to those achieved by some of the participants, all of which were either heavily-supervised or manually coded systems. It is important to bear in mind that the discovery procedure had no benefit of training material, or any information beyond the seed pattern set.

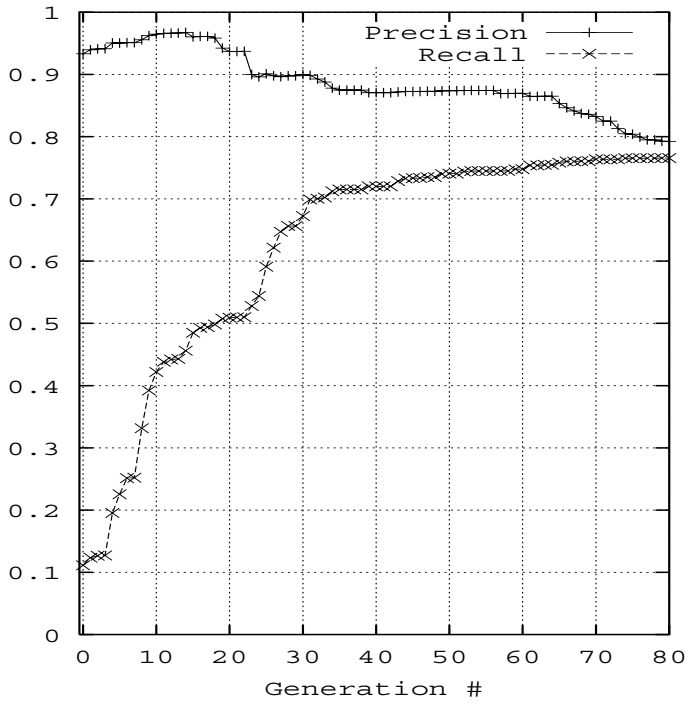

Figure 1: Recall/Precision curves for Management Succession

4.2 Choice of Test Corpus

Figure 2 shows two evaluations of our discovery procedure, tested against the original MUC-6 corpus of 100 documents, and against our test corpus, which consists of an additional 150 documents judged manually. The two plots in the figure show a slight difference in results, indicating that in some sense, the MUC corpus was more "random", or that our expanded corpus was somewhat skewed in favor of more common patterns that the system is able to find more easily.

4.3 Choice of Evaluation Metric

The graphs shown in Figures 1 and 2 are based on an "objective" measure we adopted during the experiments. This is the same measure of relevance used internally by the discovery procedure on each iteration (relative to the "truth" of relevance scores of the previous iteration), and is not quite the standard measure used for text filtering in IR. According to this measure, the system gets a score for each document based on the relevance which it assigned to the document. Thus if the system assigned relevance of X percent to a relevant document, it only received X

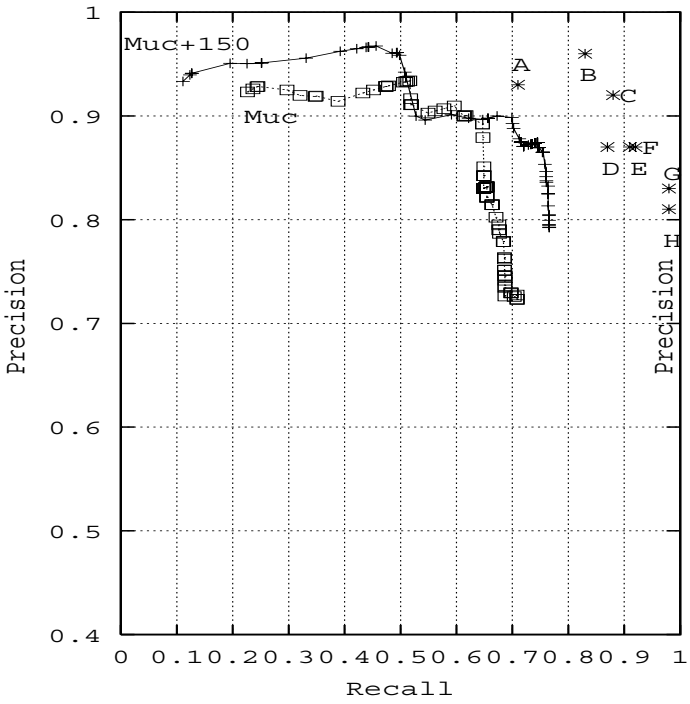

Figure 2: Precision vs. Recall

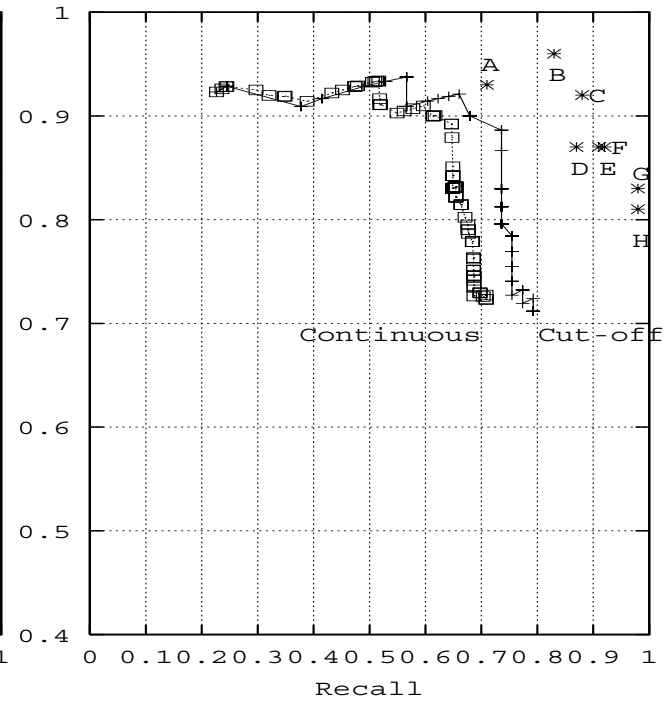

Figure 3: Results on the MUC corpus

percent on the recall score for classifying that document correctly. Similarly, if the system assigned relevance Y to an irrelevant document, it was penalized only for the mis-classified Y percent on the precision score. To make our results more comparable to those of other MUC competitors, we chose a cut-off point and force the system to make a binary relevance decision on each document. The cut-off of 0.5 seemed optimal from empirical observations. Figure 3 shows a noticeable improvement in scores, when using our continuous, "objective" measure, vs. the cut-off measure, with the entire graph essentially translated to the right for a gain of almost 10 percentage points of recall.

4.4 Evaluating Patterns

Another effective, if simple, measure of performance is how many of the patterns the procedure found, and comparing them with those used by an extraction engine which was manually constructed for the same task. Our MUC-6 system used approximately 75 clause level patterns, with 30 distinct verbal heads. In one conservative experiment, we observed that the discovery procedure found 17 of these verbs, or 57%. However, it also found at least 8 verbs the manual system lacked, which seemed relevant to the scenario:

company-bring-person-[as+officer][12]
person-come-[to+company]-[as+officer]
person-rejoin-company-[as+officer]
person-{return,continue,remain,stay}-[as+officer]
person-pursue-interest

At the risk of igniting a philosophical debate over what is or is not relevant to a scenario, we note that the first four of these verbs are evidently essential to the scenario in the strictest definition, since they imply changes of post. The next three are "staying" verbs, and are actually also needed, since higher-level inferences required in *tracking* events for long-range merging over documents, require knowledge of persons *occupying* posts, rather than only assuming or leaving them. The most curious one is "*person-pursue-interest*"; surprisingly, it too is useful, even in the strictest MUC sense, cf., (muc, 1995). Systems are judged on filling a slot called *"other-organization"*, indicating from or to which company the person came or went. This pattern is consistently used in text to indi-

[12]bracketed constituents are outside of the central SVO triplet, included here for clarity.

cate that the person left to pursue other, undisclosed interests, the knowledge of which would relieve the system from seeking other information in order to fill this slot. This is to say that here strict evaluation is elusive.

5 Discussion and Current Work

Some of the prior research has emphasized interactive tools to convert examples to extraction patterns, cf. (Yangarber and Grishman, 1997), while others have focused on methods for automatically converting a corpus annotated with extraction examples into such patterns (Lehnert et al., 1992; Fisher et al., 1995; Miller et al., 1998). These methods, however, do not reduce the burden of *finding* the examples to annotate. With either approach, the portability bottleneck is shifted from the problem of building patterns to that of finding good candidates.

The prior work most closely related to this study is (Riloff, 1996), which, along with (Riloff, 1993), seeks automatic methods for filling slots in event templates. However, the prior work differs from that presented here in several crucial respects; firstly, the prior work does not attempt to find entire events, after the fashion of MUC's highest-level scenario-template task. Rather the patterns produced by those systems identify NPs that fill *individual* slots, without specifying how these slots may be combined at a later stage into complete event templates. The present work focuses on directly discovering *event-level*, multi-slot relational patterns. Secondly, the prior work either relies on a set of documents with relevance judgements to find slot fillers where they are relevant to events, (Riloff, 1996), or utilizes an un-classified corpus containing a very high proportion of relevant documents to find all instances of a semantic class, (Riloff and Jones, 1999). By contrast, our procedure requires no relevance judgements, and works on the assumption that the corpus is balanced and the proportion of relevant documents is small. Classifying documents by hand, although admittedly easier than tagging *event* instances in text for automatic training, is still a formidable task. When we prepared the test corpus, it took 5 hours to mark 150 short documents.

The presented results indicate that our method of corpus analysis can be used to rapidly identify a large number of relevant patterns without pre-classifying a large training corpus. We are at the early stages of understanding how to optimally tune these techniques, and there are number of areas that need refinement. We are working on capturing the rich information about concept classes which is currently returned as part of our pattern discovery procedure, to build up a concept dictionary in tandem with the pattern base. We are also considering the proper selection of weights and thresholds for controlling the rankings of patterns and documents, criteria for terminating the iteration process, and for dynamic adjustments of these weights. We feel that the generalization technique in pattern discovery offers a great opportunity for combating sparseness of data, though this requires further research. Lastly, we are studying these algorithms under several unrelated scenarios to determine to what extent scenario-specific phenomena affect their performance.

References

Ido Dagan, Shaul Marcus, and Shaul Markovitch. 1993. Contextual word similarity and estimation from sparse data. In *Proceedings of the 31st Annual Meeting of the Assn. for Computational Linguistics*, pages 31–37, Columbus, OH, June.

David Fisher, Stephen Soderland, Joseph McCarthy, Fangfang Feng, and Wendy Lehnert. 1995. Description of the UMass system as used for MUC-6. In *Proc. Sixth Message Understanding Conf. (MUC-6)*, Columbia, MD, November. Morgan Kaufmann.

R. Grishman, L. Hirschman, and N.T. Nhan. 1986. Discovery procedures for sublanguage selectional patterns: Initial experiments. *Computational Linguistics*, 12(3):205–16.

Lynette Hirschman, Ralph Grishman, and Naomi Sager. 1975. Grammatically-based automatic word class formation. *Information Processing and Management*, 11(1/2):39–57.

Timo Järvinen and Pasi Tapanainen. 1997. A dependency parser for English. Technical Report TR-1, Department of General Linguistics, University of Helsinki, Finland, February.

Martin Kay and Martin Röscheisen. 1993.

Text-translation alignment. *Computational Linguistics*, 19(1).

W. Lehnert, C. Cardie, D. Fisher, J. McCarthy, E. Riloff, and S. Soderland. 1992. University of massachusetts: MUC-4 test results and analysis. In *Proc. Fourth Message Understanding Conf.*, McLean, VA, June. Morgan Kaufmann.

Scott Miller, Michael Crystal, Heidi Fox, Lance Ramshaw, Richard Schwartz, Rebecca Stone, Ralph Weischedel, and the Annotation Group. 1998. Algorithms that learn to extract information; BBN: Description of the SIFT system as used for MUC-7. In *Proc. of the Seventh Message Understanding Conference*, Fairfax, VA.

1993. *Proceedings of the Fifth Message Understanding Conference (MUC-5)*, Baltimore, MD, August. Morgan Kaufmann.

1995. *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, Columbia, MD, November. Morgan Kaufmann.

Johanna Nichols. 1978. Secondary predicates. *Proceedings of the 4th Annual Meeting of Berkeley Linguistics Society*, pages 114–127.

Maria Teresa Pazienza, editor. 1997. *Information Extraction*. Springer-Verlag, Lecture Notes in Artificial Intelligence, Rome.

Fernando Pereira, Naftali Tishby, and Lillian Lee. 1993. Distributional clustering of English words. In *Proceedings of the 31st Annual Meeting of the Assn. for Computational Linguistics*, pages 183–190, Columbus, OH, June.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, Orlando, Florida.

Ellen Riloff. 1993. Automatically constructing a dictionary for information extraction tasks. In *Proceedings of Eleventh National Conference on Artificial Intelligence (AAAI-93)*, pages 811–816. The AAAI Press/MIT Press.

Ellen Riloff. 1996. Automatically generating extraction patterns from untagged text. In *Proceedings of Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 1044–1049. The AAAI Press/MIT Press.

Pasi Tapanainen and Timo Järvinen. 1997. A non-projective dependency parser. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, pages 64–71, Washington, D.C., April. ACL.

Roman Yangarber and Ralph Grishman. 1997. Customization of information extraction systems. In Paola Velardi, editor, *International Workshop on Lexically Driven Information Extraction*, pages 1–11, Frascati, Italy, July. Università di Roma.