Learning Information Extraction Patterns from Examples

Scott B. Huffman

Price Waterhouse Technology Centre, 68 Willow Road, Menlo Park CA 94025, USA

Abstract. A growing population of users want to extract a growing variety of information from on-line texts. Unfortunately, current information extraction systems typically require experts to hand-build dictionaries of extraction patterns for each new type of information to be extracted. This paper presents a system that can learn dictionaries of extraction patterns directly from user-provided examples of texts and events to be extracted from them. The system, called LIEP, learns patterns that recognize relationships between key constituents based on local syntax. Sets of patterns learned by LIEP for a sample extraction task perform nearly at the level of a hand-built dictionary of patterns.

1 Introduction

Information extraction can be defined as the detection and extraction of particular events of interest from text. Although significant progress has been made on information extraction systems in recent years (for instance through the MUC conferences [MUC, 1992; MUC, 1993]), coding the knowledge these systems need to extract new kinds of information and events is an arduous and time-consuming process [Riloff, 1993]. The dictionaries of syntactic and semantic patterns used to recognize each type of event are typically built by hand by a team of highly-trained specialists. As the amount of on-line text (newswires, World Wide Web documents, etc.) and the number of users with access continues to grow, however, there is a need to extract an ever-widening diversity of types of information and events. Having specialists hand-build extraction knowledge for this diversity of extraction tasks is untenable.

This paper examines an alternative: machine learning of dictionaries of information extraction patterns from user-provided examples of events to be extracted. We present a system called LIEP (for Learning Information Extraction Patterns) that learns such a dictionary given example sentences and events. In a sample extraction task (extracting corporate management changes), LIEP learns sets of patterns that achieve performance comparable to a meticulously hand-built dictionary of patterns.

We will begin with a brief description of the information extraction task and the extraction technique that the system uses. Next, we will turn to the learning algorithm and present an example of its operation. Finally, we will describe the system's results on both the management changes extraction task and a corporate acquisitions extraction task, and discuss opportunities for further research.

2 The extraction task

Full-scale extraction systems such as those in the MUC contests typically include a sentence-level extraction phase, followed by a "merging" phase in which information drawn from different sentences is combined. This work focuses on learning to extract information within individual sentences. Soderland and Lehnert [1994] have described one technique for learning to perform the merging process.

Extracting an event from text typically involves recognizing a group of entities of specific types that have particular relationships between them. Entities are generally expressed as noun phrases. To recognize an event in a sentence, a system must identify the entities of interest, and determine that the syntactic and semantic relationships within the sentence indicate the event and the entities' roles in it.

The primary tasks we have applied our system to so far are extracting corporate management changes and corporate acquisitions from newswire texts. In the management changes domain (which we will use to describe the system in this paper) the entities of interest are companies, people, and management titles (e.g., "vp of finance", "CEO", etc.). A variety of syntactic and semantic relationships between these entities in a sentence can indicate a management change event.

NLP-based extraction techniques, as opposed to simple keyword, proximity, or topic/entity searches, are needed for reasonably accurate extraction for this task. Not every combination of person/company/title in close proximity indicates a management change, even when other keyword indicators (e.g., named, announced, appointed, etc.) are nearby. For instance, consider:

> NORTH STONINGTON, Connecticut (Business Wire) – 12/2/94 – Joseph M. Marino and Richard P. Mitchell have been named senior vice presidents of Analysis & Technology Inc. (NASDAQ NMS: AATI), Gary P. Bennett, president and CEO, has announced.

Here, Joseph M. Marino and Richard P. Mitchell participate in management changes, but Gary P. Bennett does not, despite the mention of both a company and a title near his name. LIEP learns patterns that correctly handle examples like this one.

3 The extraction system

The extraction system that uses the extraction patterns learned by LIEP is called ODIE (for "On-Demand Information Extractor"). ODIE processes an input text using a fairly typical set of phases for such systems (as described, e.g., by Hobbs [1993]). It is perhaps closest in design to SRI's FASTUS [Hobbs *et al.*, 1992] and UMass's CIRCUS [Lehnert *et al.*, 1993].

Given an input text, ODIE first tokenizes the text and breaks it into sentences. For each sentence, ODIE first checks whether the sentence contains any of a set of keywords that indicate the possibility that the sentence expresses an event of interest. If no keywords are found, the sentence is thrown away; otherwise, the words in the sentence are tagged with their parts of speech.[1] Next, a set of simple pattern-matchers run over the sentence to identify entities of interest (for management changes, this is people, company names, and management titles) and contiguous syntactic constituents (noun groups, verb groups, and prepositions). The grammars used for identifying noun and verb groups are loosely based on those used by FASTUS [Hobbs *et al.*, 1992].

```
n_was_named_t_by_c:
   noun-group(PNG,head(isa(person-name))),
   noun-group(TNG,head(isa(title))),
   noun-group(CNG,head(isa(company-name))),
   verb-group(VG,type(passive),head(named or elected or appointed)),
   preposition(PREP,head(of or at or by)),

   subject(PNG,VG),
   object(VG,TNG),
   post_nominal_prep(TNG,PREP),
   prep_object(PREP,CNG)
 ==> management_appointment(M,person(PNG),title(TNG),company(CNG)).
```

Fig. 1. An information extraction pattern.

Next, ODIE applies a set of information extraction patterns such as that shown in Figure 1 to identify events. Patterns match syntactic constituents by testing their head words/entities and other simple properties (e.g. `active/passive` for verb groups), and attempt to verify syntactic relationships between the constituents. If all of the syntactic relationships are verified, an event is logged. For instance, the pattern shown in Figure 1 will log a management change event in a sentence like "Sue Smith, 39, of Menlo Park, was appointed president of Foo Inc." Here, `PNG` is "Sue Smith", `VG` is "was appointed", `TNG` is "president", `PREP` is "of", and `CNG` is "Foo Inc."

Rather than construct a complete and consistent parse of the entire sentence, ODIE attempts only to verify the plausibility of specific syntactic relationships between pairs of constituents tested in extraction patterns. A relationship's plausibility is verified using local syntactic constraints. For instance, the relationship `subject(ng,vg)` is considered plausible if `ng` is directly to the left of `vg`, or if `ng` is further to the left, and everything in between `ng` and `vg` could possibly be a right-modifier of `ng` – for instance, prepositional phrases, comma-delimited strings of

[1] We are currently using Eric Brill's part-of-speech tagger [Brill, 1994].

words like relative clauses, parentheticals, etc. Similar plausibility judgments are made for other syntactic relationships like `object`, `post-nominal-preposition`, `preposition-object`, etc.[2]

Performing simple, local plausibility verifications "on demand" for only the syntactic relationships in extraction patterns can be contrasted with the "full parsing" of standard NLP systems. The advantage of on-demand parsing, of course, is avoiding the difficult, time-consuming, and semantic knowledge-intensive process of full parsing. The disadvantage is that on-demand parsing's local, non-semantic nature does not provide enough constraint; it can overgenerate. For instance, multiple noun groups can plausibly hold the `subject` relationship with a given verb group. In a full parsing system such syntactic overgeneration would be constrained by semantics. In on-demand parsing, it is constrained by tests of constituents' heads, properties, and other syntactic relationships within each extraction pattern. For instance, ODIE never generates all possible `subject` relationships, but rather checks the plausibility of `subject(ng,vg)` *only* for `ng`'s and `vg`'s that pass the other tests in a specific extraction pattern – such as the tests at the top of Figure 1 on `PNG` and `VG`. These tests rule out most `ng` and `vg` combinations before any parsing knowledge is even applied. In cases where `subject(ng,vg)` is checked and found plausible, the relationship is only "accepted" (affects the system's output) if the rest of the relationships in the pattern are also plausible. Essentially, ODIE banks on the likelihood that enough constraint to avoid most false hits comes from the combination of local syntactic relationships between constituents in specific extraction patterns.

4 Learning information extraction patterns

LIEP learns extraction patterns like that shown in Figure 1 from example texts containing events. Most previous work on learning for information extraction has used the large corpus of pre-scored training texts provided for the MUC contests as training input (e.g., [Riloff, 1993; Soderland and Lehnert, 1994]). However, since such a corpus is not available for most extraction tasks, LIEP allows a user to interactively identify events in texts. In the current system, a simple HTML-based interface is used for this. For each sentence of a training text given by the user, entities of interest (e.g. people, companies, and titles) are identified, and the user can then choose which combinations of the entities signify events to be extracted. An event (e.g., a management change) includes a set of roles (e.g., person, title, company) with specific entities filling each role. Each positive example thus consists of a sentence, processed to identify entities and syntactic constituents, and an event that occurs in the sentence.

[2] The most recent version of ODIE encodes its extraction pattern dictionary as a finite state machine. Each pattern like the one in Figure 1 takes the form of a path of transitions through the machine, rather than a separate rule; syntactic relationships like `subject` are recognized by embedded sub-machines. LIEP learns new paths that are added to this finite state machine, rather than new rules. However, the learning algorithm used is very similar to what is presented here.

LIEP tries to build a set of extraction patterns that will maximize the number of extractions of positive examples and minimize spurious extractions. Given a new example that is not already matched by a known pattern, LIEP first attempts to generalize a known pattern to cover the example. If generalization is not possible or fails to produce a high-quality pattern, LIEP attempts to build a new pattern based on the example. We will first describe how new patterns are built, and then how they are generalized.

4.1 Building new patterns

An extraction pattern matches possible role-filling constituents, and tests for a set of syntactic relationships between those constituents that, if present, indicate an event. Other constituents, such as verb groups, are included in a pattern only when needed to create relationships between the role-filling constituents. LIEP creates potential patterns from an example sentence/event by searching for sets of relationships that relate all of the role-filling constituents in the event to one another. Since our example extraction task has three constituents, LIEP attempts to find paths of relationships between each pair of constituents (three pairs) and then merges those paths to create sets of relationships relating all three.

The relationship between a pair of constituents can either be direct – as between ng and vg if `subject(ng,vg)` holds – or indirect, where the constituents are the endpoints of a path of relationships that passes through other intermediate constituents. For instance, in (1) "Bob was named CEO of Foo Inc.", Bob and CEO are related by

```
[subject(Bob,named),object(named,CEO)]
```

```
find_relationships(C1,C2) {
    if direct_relationship(C1,C2,R) then return(R)
    else
    while (choose_next_intermediate_constituent(CIntermediate)) {
        Rels1 = find_relationships(C1,CIntermediate)
        Rels2 = find_relationships(C2,CIntermediate)
        return(Rels1 + Rels2)}
    else failure.}
```

Fig. 2. Finding the path of plausible syntactic relationships between two constituents.

To find relationships between pairs of constituents, LIEP uses the recursive, depth-first algorithm shown in Figure 2. It first tries to find a direct relationship between the constituents. If there is none, it chooses another constituent in the sentence and tries to find paths of relationships between each of the original constituents and this intermediate constituent. `Choose_next_intermediate_constituent` selects intermediate constituents to use starting from the rightmost constituent between the two being related, and moving left toward the beginning of the sentence.

In some cases, there are multiple paths of relationships between a pair of constituents. The multiple paths very roughly correspond to multiple syntactic parses of the sentence. For instance, in the above sentence, "of Foo Inc." could modify the verb `named` or the noun `CEO`. Thus, `Bob` and `Foo Inc.` are related by both:

```
[subject(Bob,named),object(named,CEO),
 post_verbal_post_object_prep(named,of),
 prep_object(of,Foo Inc.)]
```

and:

```
[subject(Bob,named),object(named,CEO),
 post_nominal_prep(CEO,of),
 prep_object(of,Foo Inc.)]
```

LIEP does not reason about what "of Foo Inc." modifies; it simply generates both of the possibilities because ODIE's knowledge of plausible syntactic relationships indicates that both `post_verbal_post_object_prep(named,of)` and `post_nominal_prep(CEO,of)` are plausible.

In other cases, no path of relationships between a pair of constituents can be found. This indicates that ODIE's set of syntactic relationships (which is very simple) is insufficient to cover the example. A common example of this occurs because ODIE does not understand parallel structure, as in "Foo Inc. named Bob CEO, and *Jane president.*" ODIE cannot relate `Foo Inc.` to `Jane` or `president` because it cannot recognize their relationships to the verb. Thus LIEP cannot create a pattern from the example `company(Foo Inc.), person(Jane), title(president)` using the built-in relationships. This is not a weakness in the learning algorithm, but in the syntactic vocabulary used to analyze the examples – in machine learning terms, the representation bias.

In the most recent version of LIEP, when a path of relationships cannot be found between the constituents in an example, the system induces a new relationship that connects the smallest unrelatable gap between the constituents – essentially extending the system's vocabulary for analyzing the example. This new relationship allows LIEP to learn a pattern for the example, and the new relationship can also be used in later patterns. This ability to induce new relationships improves LIEP's overall performance, but because it is still being tested, we will not describe the capability in detail in this paper.

Figure 3 shows the routine `build_new_pattern(Example,Ptn)`. The routine `Find_relationships_between_role_fillers` calls `find_relationships` for each

```
build_new_pattern(Example) {
    HighestAccuracy = 0, Result = failure
    do 3 times {
        Rels = find_relationships_between_role_fillers(Example)
        if (Rels != failure) then {
            Pattern = create_pattern_from_relationships(Rels)
            Acc = compute_f_score_on_old_examples(Pattern)
            if Acc > HighestAccuracy then {
                HighestAccuracy = Acc
                Result = Pattern }}}
    return(Result).}
```

Fig. 3. Building a new pattern for a positive example.

pair of roles in the example event, and merges the resulting sets of relationships. Calling it multiple times causes `find_relationships` to backtrack and find multiple paths between constituents if they exist. We have arbitrarily chosen to generate up to three paths of relationships between the role-filler constituents. `Create_pattern_from_relationships` converts each path of relationships into an extraction pattern like that in Figure 1, in a straightforward way. In addition to the relationships themselves, a test is added to the pattern for each constituent appearing in the set of relationships. The test matches the constituent's head word/entity, and other syntactic properties (e.g. `active/passive`).

As an example, consider again (1) "Bob was named CEO of Foo Inc." The first set of relationships `Find_relationships_between_role_fillers` finds relating `Bob`, `CEO`, and `Foo Inc.` is:

```
[subject(Bob,named),object(named,CEO),
 post_verbal_post_object_prep(named,of),
 prep_object(of,Foo Inc.)]
```

From these, `create_pattern_from_relationships` creates the pattern:

```
LIEP_pattern1:
  noun-group(PNG,head(isa(person-name))),
  noun-group(TNG,head(isa(title))),
  noun-group(CNG,head(isa(company-name))),
  verb-group(VG,type(passive),head(named)),
  preposition(PREP,head(of)),

  subject(PNG,VG),
  object(VG,TNG),
  post_verbal_post_object_prep(VG,PREP),
  prep_object(PREP,CNG)
==> management_appointment(person(PNG),title(TNG),company(CNG)).
```

After up to three such patterns are constructed, they are compared by running them on all the example sentences LIEP has seen so far. The pattern with the highest F-measure[3] is returned and added to the system's dictionary.

4.2 Generalizing patterns

The new patterns LIEP learns are fairly specific: for non-role-filler constituents, they test for specific properties and head words (e.g., **named**). Often, later training examples have the same syntactic relationships as a previously learned pattern, but with different constituent head words or properties. This indicates that the pattern can be generalized.

LIEP assumes that non-role-filler constituents' head words and properties within a pattern can be generalized, but that constituents' syntactic types and relationships – what might be called the pattern's "syntactic footprint" – should not be generalized. For instance, if LIEP sees a future example which is similar to **LIEP_pattern1** except that **subject(PNG,VG)** is replaced with some other relationship, it will not try to generalize **LIEP_pattern1** but rather create a completely new pattern.

In order to recognize when a pattern might be generalized based on a new example, while learning a new pattern LIEP also creates a special version of the pattern that tests only its "syntactic footprint" – that is, the non-generalizable parts of the full pattern. For LIEP_pattern1, this is:

```
LIEP_pattern1(NON-GENERALIZABLE-PORTION):
  noun-group(PNG,head(isa(person-name))),
  noun-group(TNG,head(isa(title))),
  noun-group(CNG,head(isa(company-name))),
  verb-group(VG), preposition(PREP),

  subject(PNG,VG),
  object(VG,TNG),
  post_verbal_post_object_prep(VG,PREP),
  prep_object(PREP,CNG)
==> matches_positive_example(person(PNG),title(TNG),company(CNG)).
```

Consider the example (2) "Joan has been appointed vp, finance, at XYZ Company." Although it uses different head words, this example has the same syntactic relationships between its person, title, and company constituents as the previous example that resulted in **LIEP_pattern1**. LIEP notices the similarity because **LIEP_pattern1(NON-GENERALIZABLE-PORTION)** matches the new

[3] The F-measure [Chinchor and Sundheim, 1993] balances the recall and precision performance of the pattern being tested. For our tests we used $\beta = 1.0$ (recall and precision equally important).

example. The system forms a generalization of LIEP_pattern1 by inserting disjunctive values within each generalizable test in the pattern. These disjunctive values match the value(s) already in the pattern, plus the value in the new example. The generalized version of LIEP_pattern1 that results is:

```
Gen1_LIEP_pattern1:
   noun-group(PNG,head(isa(person-name))),
   noun-group(TNG,head(isa(title))),
   noun-group(CNG,head(isa(company-name))),
   verb-group(VG, type(passive), head(member(genclass1))),
   preposition(PREP, head(member(genclass2))),

   subject(PNG,VG),
   object(VG,TNG),
   post_verbal_post_object_prep(VG,PREP),
   prep_object(PREP,CNG)
 ==> management_appointment(person(PNG),title(TNG),company(CNG)).

genclass1 = (named,appointed).
genclass2 = (of,at).
```

This generalized pattern replaces the original pattern in the pattern dictionary.

Later examples can cause further generalizations (further additions to the disjunctive value sets, which LIEP calls **genclasses**). In addition, for open-class words (nouns and verbs), LIEP re-uses the genclasses it learns across patterns. For instance, if it has learned a genclass containing **named**, **appointed**, and **elected** by generalizing a pattern, when generalizing some other pattern containing **named**, it will use that genclass instead of creating a new one. For closed-class items like prepositions, LIEP always creates a new genclass for each rule, because those items are often used in a context-specific way.

Since for open-class items, what LIEP is learning is essentially a contextualized set of synonyms, a more aggressive learning strategy would be to use a synonym dictionary like WordNet [Miller, 1990] to propose possible synonyms to the user when a new pattern is first learned. This would reduce the number of training examples needed by the system. We plan to investigate this approach in future work.

5 Results

To test LIEP's performance on the management changes task, we collected a corpus of 300 naturally-occurring texts reporting management changes. The corpus was drawn from newswire articles appearing in the Wall Street Journal, New York Times, Business Wire, PR Newswire, and other newswire services, in January and early February 1995. Each corpus text contains either one or two sentences from a newswire article. Many of the corpus sentences are complex, and contain multiple names and companies. Often, more than one management

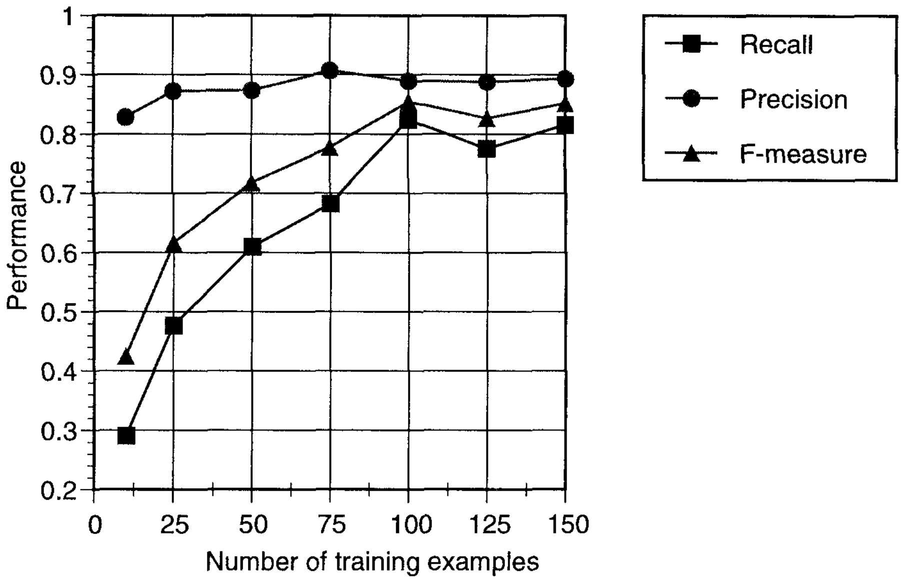

Fig. 4. LIEP's recall, precision, and F-measure on 100 test sentences per number of training examples, on the management changes extraction task.

change is reported in a single sentence, either through multiple complete clauses or parallel structure.

We ran LIEP multiple times over randomly-chosen training sets of different sizes. For each run, LIEP was trained using some number of randomly selected training texts from our 300-text corpus, and then its performance was tested on a disjoint set of 100 randomly selected test texts from the corpus. Figure 4 graphs the system's recall, precision, and F-measure on the test sets for different numbers of training examples. Each point in the graph is averaged over five runs.

ODIE's average F-measure using a hand-built set of patterns on randomly selected sets of 100 test texts from our corpus is 89.4% (recall 85.9%; precision 93.2%; averaged over ten runs). As Figure 4 shows, after 150 training texts, LIEP reaches an average F-measure of 85.2% (recall 81.6%; precision 89.4%) – a difference of less than five percent from the hand-built patterns.

Figure 5 shows the number of patterns LIEP learns and generalizes for given numbers of training texts (again, averaged over five runs). Not surprisingly, as the number of training examples increases, the number of new patterns LIEP has to learn begins to level off, and more of the learning involves generalizing previously learned patterns. Figure 6 emphasizes this point by comparing the percentage of new patterns versus generalized patterns learned. As the number of training examples increases, the percentage of learning by generalized previous patterns grows.

We have also applied LIEP to a corporate acquisitions extraction task – the

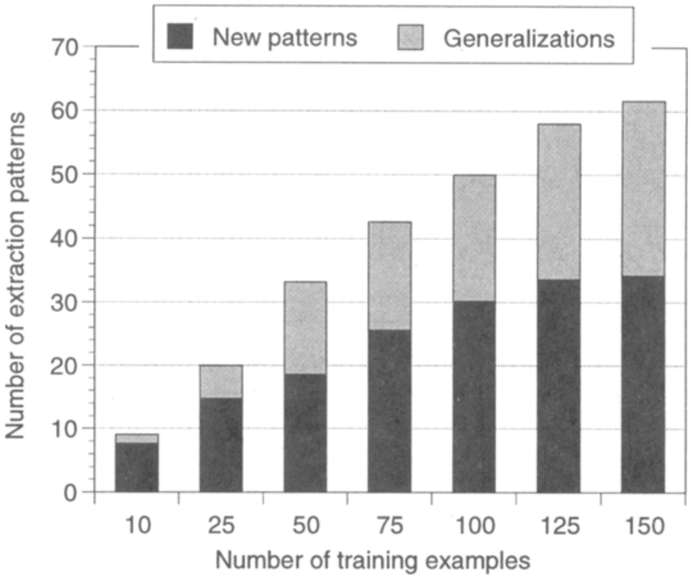

Fig. 5. Number of patterns learned and patterns generalized by LIEP on the management changes task.

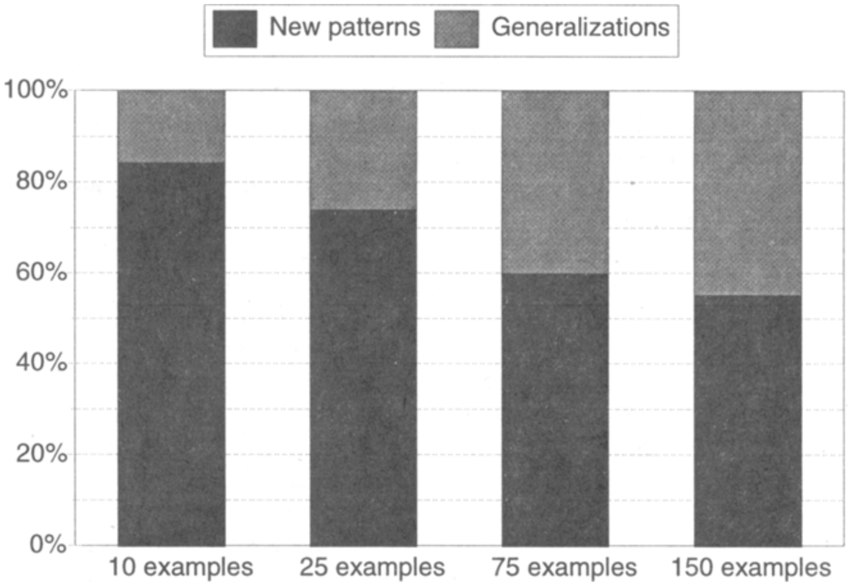

Fig. 6. Percentage of patterns learned and patterns generalized on the management changes task.

event to extract is when one company acquires part or all of another. Similar to management changes, for this task we collected 300 naturally-occurring texts reporting acquisitions from newswire articles of April-July 1995. We followed a similar train/test procedure to that used for the management changes. The resulting learning curves are quite similar to the curves in Figure 4, with the difference that acquisitions performance is a few percent lower overall. After 100 training examples, LIEP achieves an F-measure of 72.6% for acquisitions, compared to 85.4% for management changes. After 150 training examples, the acquisitions F-measure is 75.7%; after 200, it is 76.8%. These results are not surprising, because the language news articles use to report acquisitions is more variable than that used for management changes.

6 Discussion

LIEP is perhaps most closely related to Riloff's AutoSlog system [Riloff, 1993]. AutoSlog is a knowledge acquisition tool that uses a training corpus to generate proposed extraction patterns for the CIRCUS extraction system. A user either verifies or rejects each proposed pattern. AutoSlog does not try to recognize relationships between multiple constituents, as LIEP does; rather, it builds smaller patterns that recognize instances of single role-fillers. Later stages of CIRCUS then combine these instances into larger events.

Recently, Soderland *et al.* [1995] have described CRYSTAL, a system that induces extraction patterns from marked example texts. The general task is similar to LIEP's, but the specific problem and the approaches used are quite different. LIEP learns to extract events with multiple roles, and bases its learning on the relationship between those roles in each example. CRYSTAL, on the other hand, learns to extract single entities in particular categories (e.g. "pre-existing disease"). Thus, rather than relating entities, CRYSTAL's main learning task is to learn what parts of the context around an entity indicate that it is in a category. It uses a specific-to-general induction approach for this, in which it initially includes all the context in each example, and then generalizes by comparing and merging. Because the learning task is less strongly biased than LIEP's, CRYSTAL requires more training examples – on the order of thousands rather than hundreds – to achieve good performance.

One way to view LIEP's learning of new extraction patterns is as explanation-based learning [Mitchell *et al.*, 1986] with an overgeneral and incomplete domain theory. LIEP's "domain theory" is its knowledge about plausible syntactic relationships. LIEP uses this theory to explain the positive examples it is given. The theory is overgeneral in that it generates multiple explanations for some examples, and so LIEP uses an empirical process (computing F-measures on past examples) to choose between explanations. The theory is incomplete in that it cannot form an explanation (a covering set of syntactic relationships) for some examples, because the set of syntactic relationships is insufficient. A number of methods have been proposed for inductively extending domain theories to cover previously unexplainable examples (e.g., [VanLehn, 1987; Hall, 1988; Pazzani, 1991]; many others). As mentioned earlier, LIEP has recently been extended with a simple inductive method for learning new relationships when faced with extraction examples it cannot cover. In this way it extends its domain theory based on those examples, allowing it to learn a pattern dictionary with broader coverage.

Because its domain theory explains the syntactic relationships in each example but does not encompass semantics, LIEP uses the simple inductive process described in the previous section to generalize patterns for varieties of head words that express the same semantics. As mentioned previously, another extension would be to use a dictionary like WordNet as a sort of semantic "domain theory" to allow more active generalization of patterns.

In order to learn quickly (i.e. to learn new patterns from single examples), LIEP makes a strong assumption (in machine learning terms, has a strong bias) about what part of an example sentence is important for recognizing an event. In particular, it assumes that the relationships *between* the role-filling constituents will provide enough context to recognize an event. However, in some cases this assumption fails. For instance, consider this acquisition example:

"Joe Smith announced the acquisition of Foo Co. by Bar Inc."

Here, the role-filling constituents are "Foo Co." and "Bar Inc.", but the context between them provides almost no information. If given this example, LIEP would learn the highly overgeneral pattern `(company1) by (company2)`. Thus, another area of future work on LIEP is to develop a better bias on the amount of context to include from an example. We are hopeful that a fairly small set of heuristics (e.g. always include a verb, include noun groups that are right-modified by role-fillers, etc.) will suffice. In the acquisitions tests described earlier, LIEP was given the simple bias that patterns must include a verb group, to avoid learning highly overgeneral patterns.

Finally, LIEP actively learns patterns from positive examples, but uses negative examples (e.g., the combinations of person/title/company that are *not* to be extracted) only as empirical evidence when comparing positive patterns. Another possibility would be to actively use negative examples, either to specialize positive patterns and/or to learn patterns that reject extractions under certain conditions. For example, in the hand-built pattern dictionary ODIE uses, there is a pattern that rejects a management change if it is preceded or followed by a phrase like "in (non-current-year)," to avoid extracting biographical information about the past positions a new manager has held. LIEP could possibly learn such negative patterns by generalizing across multiple false hits, or even from a single false hit given some simple guidance from the user.

7 Conclusion

In the future, users of on-line text will want the ability to quickly and easily generate information extractors for new events, without having to rely on specialized programmers. LIEP is a step towards that ability. Given only example texts and the events to extract from them, LIEP combines fairly simple learning techniques to learn dictionaries of general information extraction patterns. In a sample extraction task, patterns learned by LIEP perform nearly at the level of a hand-built pattern dictionary.

Acknowledgments

Thanks to David Steier and Scott Waterman for helpful comments on earlier drafts of this chapter.

References

[Brill, 1994] E. Brill. Some advances in transformation-based part of speech tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pages 722–7. 1994.

[Chinchor and Sundheim, 1993] N. Chinchor and B. Sundheim. MUC-5 evaluation metrics. In *Proceedings of the Fifth Message Understanding Conference (MUC-5)*. Morgan Kaufmann, San Mateo, CA, 1993.

[Hall, 1988] R. J. Hall. Learning by failing to explain. *Machine Learning*, 3(1):45–77, 1988.

[Hobbs *et al.*, 1992] J. R. Hobbs, D. E. Appelt, J. S. Bear, D. J. Israel, and W. Mabry Tyson. FASTUS: A system for extracting information from natural-language text. Technical Report No. 519, SRI International, November 1992.

[Hobbs, 1993] J. R. Hobbs. The generic information extraction system. In *Proceedings of the Fifth Message Understanding Conference (MUC-5)*. Morgan Kaufmann, San Mateo, CA, 1993.

[Lehnert *et al.*, 1993] W. Lehnert, J. McCarthy, S. Soderland, E. Riloff, C. Cardie, J. Peterson, F. Feng, C. Dolan, and S. Goldman. UMass/Hughes: Description of the CIRCUS system used for MUC-5. In *Proceedings of the Fifth Message Understanding Conference (MUC-5)*. Morgan Kaufmann, San Mateo, CA, 1993.

[Miller, 1990] George Miller. Five papers on WordNet. *International Journal of Lexicography*, 3:235–312, 1990.

[Mitchell *et al.*, 1986] T. M. Mitchell, R. M. Keller, and S. T. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1, 1986.

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

[MUC, 1993] *Proceedings of the Fifth Message Understanding Conference (MUC-5)*. Morgan Kaufmann, San Mateo, CA, 1993.

[Pazzani, 1991] M. Pazzani. Learning to predict and explain: An integration of similarity-based, theory driven, and explanation-based learning. *Journal of the Learning Sciences*, 1(2):153–199, 1991.

[Riloff, 1993] E. Riloff. Automatically constructing a dictionary for information extraction tasks. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)*, pages 811–16. 1993.

[Soderland and Lehnert, 1994] S. Soderland and W. Lehnert. Wrap-Up: A trainable discourse module for information extraction. *Journal of Artificial Intelligence Research (JAIR)*, 2:131–158, 1994.

[Soderland *et al.*, 1995] S. Soderland, D. Fisher, J. Aseltine, and W. Lehnert. CRYSTAL: Inducing a conceptual dictionary. In *Proceedings of 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 1314–9. Morgan Kaufmann, 1995.

[VanLehn, 1987] K. VanLehn. Learning one subprocedure per lesson. *Artificial Intelligence*, 31(1):1–40, 1987.