Machine Learning, 39, 169–202, 2000.

Machine Learning for Information Extraction in Informal Domains

DAYNE FREITAG dayne@justresearch.com
Justsystem Pittsburgh Research Center, 4616 Henry Street, Pittsburgh, PA 15213, USA

Editor: William Cohen

Abstract. We consider the problem of learning to perform *information extraction* in domains where linguistic processing is problematic, such as Usenet posts, email, and finger plan files. In place of syntactic and semantic information, other sources of information can be used, such as term frequency, typography, formatting, and mark-up. We describe four learning approaches to this problem, each drawn from a different paradigm: a rote learner, a term-space learner based on Naive Bayes, an approach using grammatical induction, and a relational rule learner. Experiments on 14 information extraction problems defined over four diverse document collections demonstrate the effectiveness of these approaches. Finally, we describe a multistrategy approach which combines these learners and yields performance competitive with or better than the best of them. This technique is modular and flexible, and could find application in other machine learning problems.

Keywords: information extraction, multistrategy learning

1. Introduction

If I were in the market for a bargain computer, then I would benefit from a program that monitors newsgroups where computers are offered for sale until it finds a suitable one for me. The design of such a program is essentially an *information extraction* (IE) problem. We know what each document (post) says in general terms; it describes a computer. Information extraction is the problem of summarizing the essential details particular to a given document. An individual summary produced by our program will take the form of a template with typed slots, in which each slot is filled by a fragment of text from the document (e.g., type: "Pentium"; speed: "200 MHz."; disksize: "1.5Gig"; etc.).

Existing work in IE can give us some good ideas about how this agent should be constructed, but we will find large portions of it inapplicable. Most of this work assumes that we can perform syntactic and semantic processing of a document. Unfortunately, not only do we find strange, syntactically intractable constructions like news headers and user signatures in news posts, but sometimes the body of a message lacks even a single grammatical construct. How should our agent handle the "messy" text it is likely to encounter? How can it exploit whatever conventions of presentation are typical of postings for this newsgroup? More interestingly, are there general machine learning methods we can use to *train* a generic agent for use in this and similarly informal domains?

Our research addresses this question. We are interested in designing machine learning components for information extraction which are as flexible as possible, which can exploit

syntactic and semantic information when it is available, but which do not depend on its availability. Other sources of useful information include:

- Term frequency statistics
- Typography (e.g., *capitalized* and *numeric*)
- Meta-text, such as HTML tags
- Formatting and layout

We are investigating the usefulness of these kinds of information by designing learning components that exploit them. Our ultimate goal is a multistrategy learning system which can be easily ported to new domains and can exploit new kinds of structure as it is encountered.

1.1. Background

The grand challenge, in which information extraction plays a part, is the development of automatic methods for the *management* of text, rather than just its transmission, storage, or display. Efforts to meet this challenge are nearly as old as computer science itself. The decades-old discipline of *information retrieval* (IR) has developed automatic methods, typically of a statistical flavor, for indexing large document collections and classifying documents. The complementary endeavor of *natural language processing* (NLP) has sought to model human language processing—with some success, but also with a hard-won appreciation of the magnitude of the task.

The much younger field of information extraction lies somewhere in between these two older endeavors, in terms of both difficulty and emphasis. IE can be regarded as a kind of cheap, directed natural language understanding. IE assumes the existence of a set of documents from a limited domain of discourse, in which each document describes one or more entities or events that are similar to those described in other documents but that differ in the details. A prototypical example is a collection of newswire articles on Latin American terrorism; each document is presumed to describe one or more terroristic acts.[1] Also defined for a given IE task is a *template*, which is a case frame (or set of case frames) that is to hold the information contained in a single article. In the terrorism example, this template might have slots for the perpetrator, victim, and instrument of the terroristic act, as well as the date on which it occurred. An IE system designed for this problem needs to "understand" an article only enough to populate the slots in this template correctly. Most of these slots can typically be filled with fragments of text taken verbatim from the document, whence the name "information extraction."

There are many variations of the IE problem, which has evolved over the decade during which it has been recognized as a distinct endeavor. Some of this evolution can be traced in the proceedings of the Message Understanding Conference (MUC) (1992, 1993, 1995), the premier forum for research in conventional IE. In its hardest form, IE involves recognizing multiple entities in a document and identifying their relationship in the populated template. On the other end of the spectrum is the problem of generic entity recognition, in which the type of the entity is relevant, but not its particular role in the document. Recognizing company names is an example of this kind of problem. Typically, there are multiple distinct sub-tasks, such as relevancy filtering, extraction, anaphora resolution, and template

merging, which together contribute to the successful performance of a given task (Cardie, 1997). Central to any IE effort, however, is the problem of extraction, of mapping text fragments to the slots they must fill in a correctly populated template. We will use the term *information extraction* to refer to this smaller, more focused problem.

As recent research in the IE community has shown, machine learning (ML) can be of service, both in performing this fragment-to-slot mapping (Kim & Moldovan, 1995; Riloff, 1996; Soderland, 1996), and in solving associated tasks (Aone & Benett, 1996; Cardie, 1993; McCarthy & Lehnert, 1995; Riloff & Lehnert, 1994; Soderland & Lehnert, 1994). At the same time, ML researchers have become increasingly aware of the IE task as a source of interesting problems (Califf, 1998). This development is part of a general growth of interest on the part of the ML community in problems involving text, which in turn can be attributed to the growth of the Internet and the Web as a source of problem domains. In fact, some ML researchers, in pursuit of automated methods for handling certain Web and Internet problems related to data mining, have discovered their affinity to research done in IE (Doorenbos, Etzioni, & Weld, 1997; Kushmerick, 1997; Soderland, 1997). Thus, a natural meeting appears to be in progress, between IE researchers, who have discovered the utility of ML methods, and ML researchers, who, as part of an interest in data mining and textual problems, have come to appreciate IE as a source of domains to motivate the refinement of existing ML methods and the development of new ones.

The result appears to be a broadening in scope of the IE problem. Although traditional research in IE has involved domains characterized by well-formed prose, and has focused on the adaptation of NLP techniques to the IE problem, many domains for which IE solutions would be useful, such as those consisting of Web pages and Usenet postings, do not have this character. In many cases, it may be hard to analyze the syntax of such documents with existing techniques. Grammar and good style are often sacrificed for more superficial organizational resources, such as simple labels, layout patterns, and mark-up tags. It is not the case that a document from such a domain is formless; rather, standard prose is replaced by domain-specific conventions of language and layout. Researchers have shown for particular types of domain that these conventions and devices can be used to learn how to perform extraction (Califf, 1998; Doorenbos, Etzioni, & Weld 1997; Kushmerick, 1997; Soderland, 1997).

1.2. Problem definition

In this paper we explore one variation of the slot-filling problem: *Find the best unbroken fragment of text to fill a given slot in the answer template.* We call the target slot a *field* and a text fragment that correctly fills it an *instance* of this field. Note that in order to study the behavior of the algorithms we develop, we have simplified the larger problem in at least two ways:

- We consider each field-learning problem in isolation. A typical IE problem involves multiple fields. These fields interact within the text in interesting ways. The best possible approach to the template-filling problem undoubtedly considers and seeks to exploit these interactions.

- In general, a field may have multiple instantiations in a file. In most cases, we concentrate on fields which either are not instantiated or have a unique instantiation in a document. For example, a Web page describing a computer science research project at a university usually lists the names of all members (i.e., all instances of the project-member field). In contrast, the project title, although it may appear several times in the page, is presumed to have a single canonical form.

Although these are not the most general assumptions, they are nevertheless sufficient to cover a wide range of IE tasks and to motivate the development of effective algorithms.

1.3. Example domain

One of the domains with which we have experimented consists of 485 electronic seminar announcements taken from the online environment of a large university. As motivation, we imagine an intelligent agent that can monitor university bulletin boards and summarize upcoming seminars for its user. Such an agent might also attempt to infer whether a seminar is interesting, and might insert details of the seminar, when ordered to do so, into the appropriate slot of the user's electronic calendar. In our collection, these details are presented in a variety of ways, as Table 1 indicates. More often than not, they do not occur in full, grammatical sentences.

What sort of information might an IE system use to process documents in such a domain effectively? Table 2 shows two "views" of a single seminar announcement drawn from our collection, a view in which literal tokens have been replaced by typographic abstractions and the "human" view. It is an interesting exercise to attempt to locate relevant seminar details, such as talk title, speaker, and start time, in the restricted view. This exercise makes clear that, although it may not be possible to locate these details with perfect accuracy, much important information is nevertheless retained in the abstract view. It may be surprising how important layout and typographic information is in the presentation of important details.

The apparent diversity of clues useful in performing information extraction in this domain motivates our research. We seek to construct a flexible architecture which can exploit useful clues in whatever form they naturally occur. Our effort has two primary aspects:

1. *Multistrategy learning*. There is reason to believe that no single document representation is best for all IE problems. We hypothesize that, by using learners from diverse paradigms to solve different aspects of an IE problem, we can achieve better results than from any single learner.
2. *Feature engineering*. The notion from ML of a *feature set* is particularly attractive, since it holds out hope of quick adaptation to domains containing novel structures. We have designed a relational rule learner, similar in spirit to FOIL (Quinlan, 1990). One input to the learner is a set of features to be used in learning. These features are intended to capture aspects of the kind of information hinted at in Table 2.

To date, we have experimented with the use of four kinds of information: raw term statistics, typography, syntax and semantics, and HTML mark-up. Our research has involved the

Table 1. Excerpts from three seminar announcements, illustrating the range of styles and devices used to convey essential details.

```
Name: Dr. Jeffrey D. Hermes
Affiliation: Department of Autoimmune Diseases Research & Biophysical Chemistry
Merck Research Laboratories
Title: "MHC Class II: A Target for Specific Immunomodulation of the
Immune Response"
Host/e-mail: Robert Murphy, murphy@a.cfr.cmu.edu
Date: Wednesday, May 3, 1995
Time: 3:30 p.m.
Place: Mellon Institute Conference Room
Sponsor:  MERCK RESEARCH LABORATORIES
```

```
Professor John Skvoretz, U. of South Carolina, Columbia, will present
a seminar entitled "Embedded Commitment," on Thursday, May 4th from
4-5:30 in PH 223D.
```

```
                    Laura Petitto
                Department of Psychology
                  McGill University

                Thursday, May 4, 1995
                     12:00 pm
                  Baker Hall 355
```

---

development of four machine learning components, each drawn from a different paradigm, and each shown to be effective for parts of the overall IE problem. Each learner is designed to learn to identify instances of a field. In the remainder of the paper we discuss each of these learners in turn, and provide evidence of their usefulness for extraction problems defined over several different domains. In Section 2, four different learning approaches are described. In Section 3 these learners are tested on fields from three information extraction domains. Finally, in Section 4 we attempt to achieve extraction performance superior to that of any individual learner by combining their predictions.

2. Learning to perform information extraction

Information extraction is essentially a kind of text classification. Consider the problem of identifying the name of the speaker in a seminar announcement. One way to approach

Table 2. Viewing a document in terms of layout and typographic information. On the left is a seminar announcement in which alphanumeric tokens have been replaced by generic place holders. On the right is the original announcement. Note how much information is contained in whitespace usage and typographic details. Details such as seminar date, start, and end time are readily identifiable in the distorted view. With a little background knowledge, even the speaker and title can be found.

```
<9.99.9.99.99.99.99.aa99+@aaaaaa.aaa.aaa.9>
Aaaa:      aaa.aaaaaa.aaaaaaaa.aaa-aaaa
Aaaaa:     AAA Aaaaaaa
Aaaaa:     99-Aaa-99
Aaaa:      9:99 - 9:99 AA
Aaaaa:     Aaaaaa Aaaa Aaaaaaaaaa
AaaaaaAa: Aaaaaa A. Aaaaaaa aa 99-Aaa-99 aa 99:99 aaaa aaaaaa.a
Aaaaaaaa:
```

```
<0.21.3.95.14.12.11.ed47+@andrew.cmu.edu.0>
Type:      cmu.andrew.official.cmu-news
Topic:     ECE Seminar
Dates:     30-Mar-95
Time:      4:00 - 5:00 PM
Place:     Scaife Hall Auditorium
PostedBy: Edmund J. Delaney on 21-Mar-95 at 14:12 from andrew.cmu.edu
Abstract:
```

```
AAAAAAAAAAAA AAAAAAA AAA AAAAAAAAAA AA AAAAAAAA AAAAAAAAAAAA

                    AAAAA A. AAAAA, AA.A.

                      Aaaaaaaaa aaa Aaaaaaa
                     Aaaaaaa Aaaa Aaaaaaa, Aaa.
                          Aaaaaaaaaa, AA

 AAAA:  Aaaaaaaa, Aaaaa 99, 9999
 AAAA:  9:99 - 9:99 A.A.
 AAAAA:  Aaaaaa Aaaa Aaaaaaaaaa
AAAAAAAAAAAA aa 9:99 A.A.
```

```
COMPUTERIZED TESTING AND SIMULATION OF CONCRETE CONSTRUCTION

                    FARRO F. RADJY, PH.D.

                      President and Founder
                    Digital Site Systems, Inc.
                         Pittsburgh, PA

 DATE:  Thursday, March 30, 1995
 TIME:  4:00 - 5:00 P.M.
 PLACE:  Scaife Hall Auditorium
REFRESHMENTS at 3:45 P.M.
```

---

this problem is to design a component which, when presented with an arbitrary fragment, outputs *yes* if it is the speaker's name, and *no* if it is not. More generally, we can ask that it output a real number, which is its confidence that a fragment is the speaker's name. Now, armed with such a component, we can present it with every appropriately sized fragment and keep the fragment for which its confidence is highest.

Given this approach, the problem of learning to identify field instances boils down to mapping a piece of text to a real number. This is the same I/O behavior as any number of techniques developed for the information retrieval and document classification tasks. In information retrieval, for example, the central task is to rank documents in order of relevance to some query (van Rijsbergen, 1979). This typically involves assigning a real number to a document which represents its degree of relevance.

Each of the learners described in this section assumes this approach. In particular, testing is always performed at the fragment level. Note that a fragment is a subsequence of tokens *within* a document. The context of a fragment is available to a learner, though a learner may choose or be designed to ignore it.

2.1. Rote learning

Perhaps the simplest possible learning approach to the IE problem is to memorize field instances verbatim, without any contextual tokens. Presented with a novel document, this approach simply matches text fragments against its "learned" dictionary, saying "field instance" to any matching fragments and rejecting all others. We can say a test fragment "matches" a dictionary entry if it contains the same tokens in the same order, modulo capitalization.

More generally, we can estimate the probability that the matched fragment is indeed a field instance. The dictionary learner we experiment with here, which we call ROTE, does exactly this. For each text fragment in its dictionary, we count the number of times it appears as a field instance (p) and the total number of occurrences in any context (t). Its confidence in a prediction is then the value p/t, smoothed for overall frequency. We use Laplace estimates under the assumption that this is a two-class problem, *field instance* and *non-field instance*. Thus, the actual confidence ROTE outputs is $(p+1)/(t+2)$.

This approach, simple as it is, is nevertheless surprisingly applicable in a wide variety of domains. Of course, its applicability depends on the nature of the problem, but at the very least a ROTE prediction, especially a high-confidence one, is a valuable piece of information. Because of the simplicity of the assumptions that go into a prediction, confidence correlates well with actual probability of correctness.

2.2. Term-space learning

We base our term-space learner on the Naive Bayes class of algorithms as typically used in document classification. Consider the problem of identifying the speaker in a seminar announcement. In this case a hypothesis takes the form, "the text fragment starting at token i and consisting of k tokens is the speaker." (Call this hypothesis $H_{i,k}$.) In the case of a seminar

announcement file, for example, $H_{309,2}$ might represent our expectation that a speaker field consists of the 2 tokens starting with the 309th token.

Bayes' Rule tells us how to combine prior expectations ($\Pr(H_{i,k})$) and current evidence ($\Pr(D \mid H_{i,k})$) to form an estimate of field membership:

$$\Pr(H_{i,k} \mid D) = \frac{\Pr(D \mid H_{i,k}) \Pr(H_{i,k})}{\Pr(D)} \tag{1}$$

In our implementation, we assume that the two components that a hypothesis comprises, position and length, are independent, and we estimate them separately:

$$\Pr(H_{i,k}) = \Pr(\text{position} = i) \Pr(\text{length} = k) \tag{2}$$

In a typical IE problem field instances are short and do not vary too much in length. Thus, simply tabulating the number of times instances of length k are seen during training, and dividing this number by the total number of training instances, yields a good estimate for the length prior.

In order to estimate the position prior, we sort training instances into n bins, based on their position, where n is much smaller than the typical document size. During testing, we use a frequency polygon drawn over these bin counts to estimate the position prior. In the experiments reported here, bins are of size 20. Training instances beginning at indexes 0 through 19 are sorted into the first bin, instances at indexes 20 through 39 are sorted into the second bin, and so on for instances at later positions. This tabulation yields a positional histogram. Position estimates for test instances located at bin midpoints are estimated directly from these counts; estimates for instances falling between bin midpoints are estimated by linear interpolation between midpoints.

The data likelihood ($\Pr(D \mid H_{i,k})$) is a product of three separate estimates:

$$\Pr(D \mid H_{i,k}) = \Pr(\text{before} \mid H_{i,k}) \Pr(\text{in} \mid H_{i,k}) \Pr(\text{after} \mid H_{i,k}) \tag{3}$$

Here, "before," "in," and "after" are shorthand for the events in which we observe the given tokens occurring before, in, and after a candidate instance, respectively. The probability $\Pr(\text{before} \mid H_{i,k})$ is a product of individual term probabilities:

$$\Pr(\text{before} \mid H_{i,k}) = \prod_{j=1}^{w} \frac{B_j\left(t_{(i-j)}\right)}{G\left(t_{(i-j)}\right)} \tag{4}$$

where w is the pre-specified context window size ($w = 4$ in these experiments), t_i is term number i in the document, $B_i(t)$ is a count of the number of times the term t occurred in the ith slot before field instances in training, and $G(t)$ is a count of the number of times term t occurred anywhere in documents in the training collection. The fraction calculated within the product is smoothed using m-estimates. Calculations for $\Pr(\text{after} \mid H_{i,k})$ are handled analogously. Calculations for $\Pr(\text{in} \mid H_{i,k})$ are similar, only normalized for length. Rather than a product, we take the average of the in-field term probabilities and multiply them by the average length of field instances. This prevents short fragments from being preferred over long ones.

Note that a true Bayesian approach, instead of $G(t)$, would have the number of training instances in the denominator. Our reason for deviating is strictly empirical. We found that the formula used works substantially better than a more principled approach (Freitag, 1999). We can justify it as a heuristic by drawing parallels to TFIDF. Using the overall frequency of a term t in the denominator has an effect similar to using the document frequency of a word in an application of TFIDF.

Unlike ROTE, this algorithm, which we will call BAYES, offers an estimate for *any* fragment. Because we want learners to reject all fragments in a document if a field is not instantiated in it, this is undesirable. The final step in training, therefore, involves the setting of a confidence threshold, below which BAYES will reject a fragment. Using its learned classifier, BAYES computes its confidence for all field instances in the training set. The threshold is a user-controlled parameter (call it γ) times the lowest field instance confidence (expressed as a log probability). Thus, if γ is set to 1.0, BAYES extracts only fragments for which it is at least as confident as some training field instance.

The settings we used in the experiments described in Section 3 for BAYES and the BAYES-grammatical inference hybrid described in the next section were based on early experiments in the seminar announcement domain. A window size of 4 was used. For m-estimates we set m to 10. BAYES's confidence threshold parameter γ was set to 2.0. This setting strongly favors recall over precision, but we take pains to adopt performance measures, such as peak F1, that normalize BAYES's overemphasis on recall.

2.3. *Learning abstract structure with grammatical inference*

Although BAYES is surprisingly good at identifying field instances, it can have difficulty identifying boundaries precisely, particularly when faced with low-frequency tokens. Often, BAYES extracts an unintelligible piece of an instance, or a fragment containing tokens from the surrounding text which a human would consider trivial to filter out (see Table 3).

BAYES's alignment difficulty is greatest for a field that typically contains tokens drawn from a large set of candidates (e.g., the speaker field; the names of people exhibit a wide variety), since the low frequency of a token's occurrence in the training set renders BAYES's probability estimates less reliable. It also has trouble extracting fields which tend to be surrounded by linguistic clues, i.e., embedded in grammatical text, rather than predictable labels or highly stereotypic language.

It is hard to avoid the impression that a simple notion of the appropriate form of a field might improve the alignment of the fields shown in Table 3. In no case in the training set, for instance, is there a seminar speaker whose name consists of two letters terminated by a period (i.e., "`Dr.`"), and we might expect a learning system to recognize that the fragment `Dr.` is always followed by additional tokens. Similarly, almost without exception, names do not contain numerals, while seminar locations usually do. It is this hole in BAYES's ability that we hope to fill with *grammatical inference* (GI). Our solution is to add a term to the product that constitutes BAYES's estimate for a fragment:

$$\Pr(H \mid D) = \Pr(\text{position}) \Pr(\text{length}) \Pr(\text{terms}) \Pr(\text{structure}) \tag{5}$$

Table 3. Examples of poor alignment from actual tests of BAYES. Each fragment is BAYES's top prediction for some document.

Location	
Confidence:	-89.69
Fragment:	GSIA 259 Refreshments served
Confidence:	-77.30
Fragment:	Mellon Institute.
Speaker	
Confidence:	-68.97
Fragment:	Dr.
Confidence:	-80.84
Fragment:	Antal Bejczy Lecture Nov. 11

Note that the *position*, *length*, and *terms* components together make up BAYES. BAYES is run exactly as described in the previous section, but its estimate for a fragment is multiplied with the new term, one that estimates the likelihood that the fragment has the right structure. This structure estimate is produced by a probabilistic learned grammar for the field in question.

2.3.1. Alergia. In broad terms, the grammatical inference problem is this: Given a set of sequences from some formal language, induce a grammar for the language. A common approach to the problem of learning *regular* grammars (i.e., finite state automata) is to construct a maximally specific grammar (called the *canonical acceptor*) which accepts only the input sequences and generalize it by merging the states in this grammar. How this merging is conducted is a detail of the particular algorithm.

In our experiments we employ ALERGIA (Carrasco & Oncina, 1994). Given the canonical acceptor, ALERGIA makes one pass through the set of pairs of states, asking for each pair, X and Y, whether X and Y are functionally equivalent. If two states are deemed equivalent, they are merged. Incoming transitions are set to point to the new, merged state. Pairs of outgoing transitions with the same label are combined, and the states they point to are also merged. This process continues recursively until all descendants of the initial pair of states are visited.

State equivalence depends on a comparison of the behaviors of the respective states. Because the automaton processed by ALERGIA is stochastic, a probability is associated with each action possible from a state. Two states are deemed equivalent if each state assigns the same probability to each possible action:

- They accept (i.e., terminate) with the same probability.

- The two transitions passing out of the states for any given symbol have the same probability, *and* the states arrived at by following these transitions are also equivalent.

ALERGIA keeps track of these probabilities by means of frequency statistics. The following counts, taken from the training sequences, are kept for each state:

- the number of times it was visited
- the number of times it accepted (was the terminal state)
- for each outgoing transition, the number of times it was followed

Whenever two states are merged during training, each count in the new state is the sum of the corresponding counts in the original states.

Of course, probabilities estimated from these counts will rarely satisfy the criteria for state equivalence because of statistical uncertainties due to finite data. ALERGIA uses Hoeffding bounds, a statistical test used to determine whether the observed behavior of a Bernoulli variable agrees with some given probability, to soften the equivalence requirement. Suppose States 1 and 2 have been visited n_1 and n_2 times, and accepted (terminated) f_1 and f_2 times, respectively. The two states are judged *compatible*, in terms of their acceptance behavior, if:

$$\left|\frac{f_1}{n_1} - \frac{f_2}{n_2}\right| < \sqrt{\frac{1}{2}\log\frac{2}{\alpha}}\left(\frac{1}{\sqrt{n_1}} + \frac{1}{\sqrt{n_2}}\right) \tag{6}$$

The same test is used to compare pairs of transitions. The α parameter controls the certainty of the equivalence judgment. Lower values of α cause more states to be judged equivalent and result in more aggressive generalization. Thus, α is a "knob" which must be set before training can occur. Successful inference depends on choosing an appropriate value.

It is straightforward to use the various counts contained in the eventual grammar to generate a probability that a test string belongs to the language encoded by the automaton. It is this probability, produced by an automaton learned over field instances, which we use for Pr(structure) in Eq. (5).

2.3.2. Inferring transducers. Before we can apply grammatical inference to this problem, we must settle on a representation for field instances, an *alphabet* to be used in labeling state transitions. One possibility would be to regard a field instance as a sequence of ASCII characters, perhaps allowing generalization to exploit some abstract character classes, as do Goan, Benson, and Etzioni (1996) This would result in a relatively large alphabet and long sequences, and would probably require a large amount of data to permit effective generalization. What is more, it would obscure certain structural features of tokens, such as whether they are capitalized, which are not apparent at the character level. Another possibility is to use the unprocessed output of the tokenizer used by BAYES and ROTE. This should result in reasonable behavior but low coverage; in fact, it would amount to a variation of ROTE. It is more important to obtain high coverage, however, since for any fragment we want some estimate of the appropriateness of its structure for the field we are trying to learn.

A more interesting idea, therefore, is to replace tokens with symbols that correspond to abstract token features, where abstraction is controlled by the structure of the field in question. For example, we would like to transduce the seminar speaker fragment, "`Dr. Koltanowski`," to something like

$$[\text{token} = \text{Dr}, \text{token} = ., \text{capitalized} = \text{true}]$$

i.e., a three-symbol sequence that effectively retains important, high-frequency tokens, but which replaces low-frequency tokens with abstractions that are relevant to the speaker field. Under this scheme, a field instance consisting of five tokens becomes a sequence of five symbols, each symbol the canonical representation of the corresponding token. We will call a procedure that transforms text in this way an *alphabet transducer*.

Because we want transduction to adapt to features of the field to be learned, we take a learning approach. We make the simplifying assumption that the right abstraction can be chosen for each token in isolation, i.e., that we need not consider its context in deciding how to represent it. For each field we construct a *decision list*, which is a sequence of pattern/emission rules. Table 4 shows part of a sample list for the location field. To transduce a token, we compare it with each pattern in turn until a matching one is found. The token is then replaced with the corresponding symbol. If no matching pattern is found, the token is replaced with the symbol `unknown`. For example, given the word "Hall", the list in Table 4 emits the symbol `word+hall`; given "5409", it emits `quad-digit+true`; and given "Baker", it emits `capitalized+true`.

We construct decision lists greedily, using a covering approach. One input to the procedure is a set of features to consider in forming patterns. In the experiments reported here, we used the 26 features shown in Table 5, all of which are readily computable by direct inspection of a token. Next, all field tokens in the set of training documents are collected into a working set of examples. At each step, a feature-value pattern that matches some of the tokens is appended to the end of the list, and all matching tokens are removed from our working set. This process repeats until either too few tokens remain in the working set or the decision list has reached a pre-specified length.

How should we choose the patterns to add to our list? Should we prefer a large list (alphabet) or a small one? Is it important that the patterns used are those that tend to distinguish field tokens from non-field tokens? Or is it sufficient to choose patterns which

Table 4. Excerpt from one decision list inferred for the location field.

If	*Emit*
word = "wean"	`word+wean`
word = "hall"	`word+hall`
triple-digit = true	`triple-digit+true`
quad-digit = true	`quad-digit+true`
capitalized = true	`capitalized+true`
...	

Table 5. The features used in the seminar announcement experiments, both for inferring alphabet transducers and training SRV. The word feature returns the literal token, modulo capitalization. Here, long means longer than four characters in length.

word	single_digit_p	long_char_p	punctuationp
singletonp	double_char_p	long_digit_p	hybrid_anum_p
doubletonp	double_digit_p	capitalized_p	a_then_num_p
tripletonp	triple_char_p	all_upper_case_p	num_then_a_p
quadrupletonp	triple_digit_p	all_lower_case_p	multi_word_cap_p
longp	quadruple_char_p	numericp	
single_char_p	quadruple_digit_p	sentence_punct_p	

distribute field tokens evenly? We experiment with three basic approaches:

- *Spread evenly.* We decide how many symbols (k) we want in our alphabet prior to inference of the decision list. At each step i, choose the feature-value test that comes closest to matching $1/(k-i-1)$ of the remaining field tokens.
- *Information gain.* Non-field tokens are also counted. Choose the feature-value pair that maximizes $p(\log(p/n) - \log(f/t))$, where p is the number of matching field tokens, n is the number of matching tokens overall, f is the number of field tokens remaining, and t is the total number of uncovered tokens overall.
- *M-estimates.* Non-field tokens are again counted. Choose the feature-value pair that maximizes $\frac{p+m/t}{n+m}$, for a small value of m ($m = 3$ in these experiments).

The results of these experiments, as well as experiments with ALERGIA's generalization parameter α, are presented in Section 3.5.

2.4. Relational learning for information extraction

Experimental results will show that the addition of grammatical inference to BAYES can yield substantial improvements. Nevertheless, the approach leaves a few things to be desired:

- Although the combination of BAYES and GI works, the way in which the two learners are combined is somewhat arbitrary. There is no guarantee that simply dropping the language-membership estimate of the GI learner into BAYES's large product assigns it its proper weight in the overall estimate.
- GI must express its patterns in terms of all tokens in a field. It may be the case, however, that only certain aspects of some tokens are important. If this is so, then the requirement that all tokens are accounted for could hamper generalization.
- Training occurs in two phases, inferring a transducer and building a grammar. Thus, the decision about which token features to use is decoupled from the process of building a classifier to recognize field instances.
- Although BAYES has a notion of field context, the GI method we use here does not. Thus, any interesting abstract patterns in contextual tokens are lost.

In summary, the use of abstract token features ought to play a direct part in the construction of a classifier to recognize field instances.

In order to meet this requirement, we look to the family of *symbolic* inductive learners, which includes decision tree learners, such as C4.5 (Quinlan, 1993), covering algorithms, such as AQ (Michalski, 1983) and CN2 (Clark & Boswell, 1991), and inductive logic programming (ILP) or *relational* learners, such as FOIL (Quinlan, 1990). A learner in this family typically takes two kinds of input: a representation language and a set of examples to be used in training. These examples are explicitly or implicitly divided into n classes. The goal of the learner is to produce a set of logical rules (or their functional equivalent) to classify a novel example into one of these classes.

Covering algorithms are designed to produce sets of rules for binary (positive vs. negative) classification problems. The outer loop of a covering algorithm repeatedly learns a rule, each iteration producing a rule that "covers" part of the training set not covered by previous rules. As each rule is produced, positive examples of the target class which are covered by the rule are removed from consideration. The process repeats with the remaining positive examples until the set of positive examples is exhausted.

The details of individual rule construction vary with the algorithm. The learner to be presented in this section, SRV, is a *top-down, greedy* rule learner. Learners in this class initialize a rule with an empty list of antecedents, such that *all* examples satisfy the rule, and iteratively add conjunctive tests, at each step choosing the test that maximizes an objective function. The objective function is designed to ensure that the learner will choose tests that include many positive examples and exclude many negative ones, while maintaining good overall coverage. This process of adding to the list of antecedents continues until some stopping criterion is satisfied. One simple criterion is that the rule match only positive examples.

Tests in a propositional covering algorithm are typically limited to asserting that a feature has some value. In *relational learning* (i.e., inductive logic programming), the set of features which define examples is augmented with a *background theory*, which defines how these features are related to each other, and to other, more abstract concepts. In addition to positing a feature-value test while constructing a classification rule, a relational learner can also derive "new" features from its background theory. This facility seems appropriate for the IE problem, where examples (field instances) are embedded in a larger structure and implicitly related to the text that surrounds them in a number of ways (e.g., contiguity and linguistic syntax). While it is not a general-purpose relational learner, SRV is designed to exploit such relational structure in a way specific to the information extraction problem.

2.4.1. Example space. Covering algorithms, as noted, require that a problem space be cast as a set of examples, positive and negative. A *positive* example for SRV is simply an instance of the target field. Any other unbroken sequence of tokens is potentially a negative example. We assume that only fragments of appropriate size—fragments having no fewer tokens than the smallest training field instance, and no more than the largest—are candidate instances. All non-instance fragments from the training documents that meet this criterion constitute the set of negative examples. Table 6 shows how the two example sets are constructed from a hypothetical seminar announcement excerpt.

Table 6. A text fragment and some of the examples it generates—one positive example, and many negative ones. How many negative examples are generated starting with any given token depends on the range in lengths of training field instances.

Fragment:	`... will meet in [Baker Hall 300], at 3:30 ...`
Positive example:	`Baker Hall 300`
Negative examples:	`will meet`
	`will meet in`
	...
	`meet in`
	`meet in Baker`
	...
	`Baker Hall 300,`
	`Baker Hall 300, at`
	...
	Hall 300
	...

Note that, even with the fragment-size heuristic, the set of negative examples is likely to be very large, typically several orders of magnitude larger than the set of positive examples. And each such fragment that satisfies the antecedents in the rule under construction is examined as part of the search for the next literal.[2] The search is usually not as expensive as it might appear, however, since the fact that fragments overlap so densely makes some efficient shortcuts possible in common situations. For simplicity of presentation, we will assume that examples are explicitly enumerated, but much more efficient implementations are in fact possible.

2.4.2. Features. In a traditional covering algorithm, such as CN2, features are all defined with respect to examples. If our object is to classify animals, for example, we use features of individual animals, such as has_spine? and lives_in_water?. Unfortunately, multi-term text fragments are difficult to describe in terms of simple features. In contrast, individual terms (or tokens), lend themselves readily to feature design. Given a token drawn from a document, a number of obvious feature types suggest themselves, such as length (e.g., single-character-word), character type (e.g., numeric), typography (e.g., capitalized), part of speech (e.g., verb), and lexical content (e.g., geographical-place). Thus, as with the transducer inferring algorithms presented in Section 2.3, SRV takes as input features defined over tokens. For experiments with the seminar announcements, we provided SRV with exactly the same 26 features used in the GI setting (Table 5).

To these 26 features we add two additional features of a different character. Whereas the typical SRV feature is a function which takes a token and returns a categorical (typically Boolean) value, the features prev_token and next_token correspond to functions which take

a token and return another token. We call the former features *simple* features; the latter are *relational* features. These two relational features only capture contiguity, but it is not difficult to see how one might implement syntactic relations (e.g., subject_verb).

2.4.3. Rule construction. SRV proceeds "top down," starting with null rules that cover the entire set of examples—all negative examples and any positive examples not covered by already induced rules—and adding literals greedily, attempting thereby to "cover" as many positive examples as possible, while weeding out covered negative examples. There are five predicates which SRV can use.

- length(Relop, N): The number of tokens in a fragment is less than, greater than, or equal to some integer. For example, the assertion length(>, 2) restricts attention to fragments containing three or more tokens.
- some(Var, Path, Feat, Value): A token in the sequence is constrained to pass a feature-value test. For example, some(?A, [ ], capitalizedp, true) means "the fragment contains some token that is capitalized." One argument to this predicate is a variable. Each distinct variable in a rule must bind to a distinct token in a fragment. The Path argument is used to exploit relational structure in the domain. Its use is described below.
- every(Feat, Value): Every token in a fragment must pass some feature-value test. For example, every(numericp, false) means "every token in the fragment is non-numeric."
- position(Var, From, Relop, N): A token bound by a some-literal in the current rule must be positioned the specified distance from either the beginning or end of the sequence. For example, position(?A, fromfirst, <, 2) means "the token bound to ?A by one or more some-literals in the current rule is either the first or second token in the fragment."
- relpos(Var1, Var2, Relop, N): Two tokens bound by distinct variables in the current rule occur in the specified order and with the specified separation. For example, relpos(?A, ?B, =, 1) means "the token bound to ?A immediately precedes the token bound to ?B."

Relational features are used only in the Path argument to the some-predicate. This argument can be empty, in which case a some-literal is asserting a feature-value test for a token actually occurring within a field, or it can be a list of relational features, in which case it is positing both a relationship between a field token and some other nearby token, as well as a feature-value for the other token. For instance, the assertion

some(?A, [ ], capitalizedp, true)

means "the fragment contains some token (?A) that is capitalized." In contrast, the assertion

some(?A, [prev_token prev_token], capitalizedp, true)

means "There is some token in the fragment preceded by a capitalized token two tokens back." There is no limit to the number of relational features the learner can string together in this way. Thus, it is possible in principle for the learner to exploit relations between tokens quite distant from each other.

The most important function of the path argument is to allow the learner to expand its consideration of relevant field structure to the text outside the field instance. These relational paths are powerful, but also potentially very costly in terms of search effort. We limit the cost heuristically. SRV works with a set of paths it is allowed to consider in creating some-literals. For each rule, this set initially contains only the null-path and all paths of length one (e.g., [prev_token]). Whenever SRV actually uses a path by adding a some-literal containing it to the rule under construction, it adds to this working set all paths created by appending any single relational feature to the path it used. In effect, SRV grows its consideration of surrounding context from field instances outward. The rate of growth is governed by how useful it has already found context to be, and how little use it is able to make of structure within the field.

When deciding which literal to add to the rule under construction, SRV uses the same gain metric as FOIL. In describing FOIL, Quinlan (1990) defines a function which characterizes the information contained in the ratio of positive to negative examples in a set of examples:

$$I(S) = -\frac{\log_2(P(S))}{(P(S) + N(S))} \tag{7}$$

where $P(S)$ and $N(S)$ are the number of positive and negative examples, respectively, in the set S. Suppose S corresponds to a set of examples covered by some partially formed rule, and let S_A stand for the subset of S covered by the rule when the literal A is added. Then the gain metric used in SRV and FOIL is:

$$\text{Gain}(A) = P(S_A)(I(S) - I(S_A)) \tag{8}$$

In order to calculate the gain of candidate literals, the entire set of examples matching the current rule must be examined. In practice, this entails loading and scanning each document in the training set. A hash table is used to account for candidate literals; with each such literal is associated a record holding two numbers, the number of positive and negative examples the literal matches. For each currently covered example in the training set, SRV updates the entry of every possible literal that matches the example, incrementing either the positive or negative count, depending on the example's label. After processing the training documents and tabulating these numbers for each candidate literal, SRV scans the hash table to select one literal maximizing the gain criterion.

Construction of a rule stops if it achieves purity—if it matches only positive examples. Often, however, especially after easy patterns have been captured by earlier rules in a training session, it may be difficult or impossible for a rule learner to reach purity. In such cases, in an effort to weed out negative examples, a rule learner may induce rules that cover very few positive examples. Not only is the probable lack of generality of such rules undesirable, but they consume a lot of search effort without advancing the learner's cause much. In order to avoid this situation, SRV discards any literal that covers fewer than five positive examples.

2.4.4. Rule accuracy estimation. The final training step is rule validation. A randomly selected portion (one-third, in this case) of the training data is set aside for validation prior to training. After training on the remaining data, the number of matches and correct

predictions is tabulated for each learned rule, and these numbers are used to estimate its accuracy. In order to get as much out of the training data as possible, this procedure—training and validation—is repeated three times, once for each of three partitions of the training data. The resulting validated rule sets are merged into a single rule set, which is used for prediction.

At testing time, associated with each rule are two counts: the number of times it matched any fragment in the validation set, and the number of times it *correctly* matched. The rule's confidence is an *m*-estimate, based on these two numbers, of the probability that the rule is correct, given that it matches a fragment. All candidate fragments in a test document are compared with all rules to look for a match. If at least one matching rule is found, SRV predicts that the fragment is a field instance and returns a confidence with this prediction. This confidence is a combination of the confidences of *all* matching rules. Suppose C is this set of confidence scores. The combined confidence is:

$$C_{\text{combined}} = 1 - \prod_{c \in C} (1 - c) \tag{9}$$

In other words, the rules are treated as independent predictors. Although this independence assumption is problematic, given the way in which validation is performed, we have found that this approach works better than, say, taking the confidence of the highest-accuracy rule.

3. Experiments

To gauge the performance of the four learners, we experimented with a number of field extraction problems defined for several different domains.

3.1. Domains

Three document collections and four problem domains formed the basis of the individual extraction tasks addressed in these experiments. The three collections from which documents were drawn differ widely in terms of the purpose of individual documents and the regularity of their structure.

The seminar announcement collection consists of 485 electronic bulletin board postings distributed in the local environment at Carnegie Mellon University. The purpose of each document in this collection is to announce or relate details of an upcoming talk or seminar. Announcements follow no prescribed pattern; documents are free-form Usenet-style postings. We annotated these documents for four fields: speaker, the name of a seminar's speaker; location, the location (i.e., room and number) of the seminar; stime, the start time; and etime, the end time.

The acquisitions collection consists of 600 documents belonging to the "acquisition" class in the Reuters corpus (Lewis, 1992). These are newswire articles that describe a corporate merger or acquisition at some stage of completion. We defined a total of ten fields for this collection, of which five are used for the experiments in this section: acquired, the official name of the company or resource in the process of being acquired; purchaser, the official

name of the purchaser; acqabr, the short form of acquired, as typically used in the body of the article (e.g., "IBM" for "International Business Machines Inc"); dlramt, the amount paid for the acquisition; and status, a short phrase indicating the status of negotiations.

The university Web page collection is a sample of pages from a large collection of university pages assembled by the World Wide Knowledge Base project (WebKB) (Craven et al., 1998). As part of an effort to classify Web pages automatically, WebKB manually assigned each of several thousand pages downloaded from four major university computer science departments to one of six classes: student, faculty, course, research project, departmental home page, and "other." From this collection we created two sub-domains, one consisting of 105 course pages, the other of 96 research project pages. The course pages were tagged for three fields: crsNumber, the official number of a course, as assigned by the university (e.g., "CS 534"); crsTitle, the official title of the course; and crsInst, the names of course instructors and teaching assistants. The project pages were tagged for two fields: projTitle, the title of the research project; and projMember, the names of the project's members.

3.2. Framework

The same experimental procedure was followed for each domain. We generated five random partitions of each collection into training and testing sets of equal size. Given a particular field, each learner was trained and tested five times, once for each partition. Reported scores represent performance over the union of the five test partitions.

3.3. Basic results

We measure the performance of learners in terms of two complementary metrics, precision and recall. A learner is said to have handled a document correctly when its most confident prediction for the document exactly identifies a field instance.[3] *Precision* is the number of correct predictions divided by the number of documents for which a learner offered some prediction. *Recall* is the number of correct predictions divided by the number of documents that have at least one field instance.

Note that these definitions regard the document as the unit of performance. This is appropriate when all instances of a field in a document refer to the same underlying entity, as is the case for most of these fields. For example, although the name of a seminar's speaker may occur several times in a document, and may have several distinct forms (e.g., "John Doe" and "Prof. John Doe"), each form refers to the same person and is assumed to be equally useful as the result of an extraction.

Two of the fields, crsInst and projMember, do not have this characteristic. We expect the search for members in a research project page to return multiple answers, each corresponding to a different person. For these two fields the unit of performance measurement is the individual prediction/field instance. The number of correct predictions is the number of predictions over all documents that exactly identified a field instance. *Precision* is the number of correct predictions divided by the total number of predictions made by a learner. *Recall* is the number of correct predictions divided by the total number of field instances in the test corpus.

Because precision and recall measure complementary aspects of a learner's performance, it is misleading to present either result without the other. What is more, because the learners described here are designed to produce a confidence with each prediction, any single precision/recall result is only part of the story. By applying a confidence threshold to a learner's predictions and ignoring all predictions having lower confidence, we can produce a learner that achieves lower recall and, typically, higher precision. In other words, we can often trade recall for precision, or vice versa.

A performance measure frequently encountered in information retrieval and information extraction is the F-measure (van Rijsbergen, 1979), which allows researchers to report a single number that combines precision and recall:

$$F = \frac{(\beta^2 + 1.0)PR}{(\beta^2 P) + R} \tag{10}$$

The parameter β determines how much to favor recall over precision. It is typical for researchers in information extraction to report the F1 score of a system ($\beta = 1$), which weights precision and recall equally. We use the *peak F1* score as the primary metric with which to compare learners. Figures 1 through 3 compare the four learners according to their peak F1 performance in the three domains. To generate this score, precision is sampled at 1% recall intervals and at full recall. The number reported is simply the maximum of the F1 scores calculated in this way. It is important to report the peak F1 score, especially for the two learners based on BAYES, because their prediction thresholds are set to low

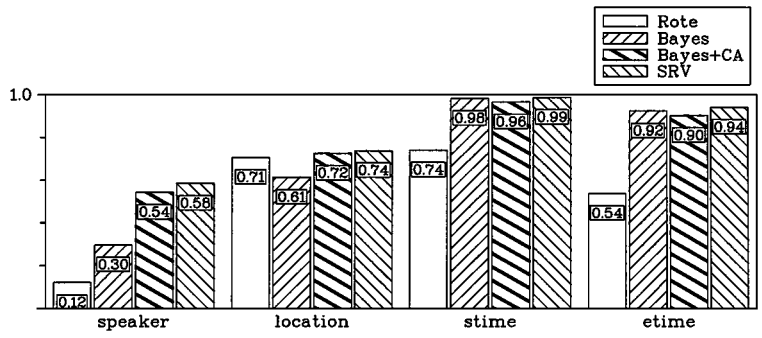

Figure 1. Peak F1 scores of the four learners on the seminar announcement fields.

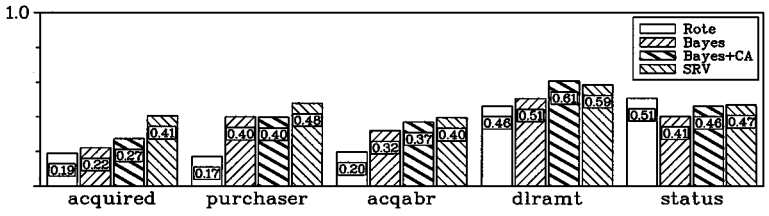

Figure 2. Peak F1 scores of the four learners on five fields from the acquisitions domain.

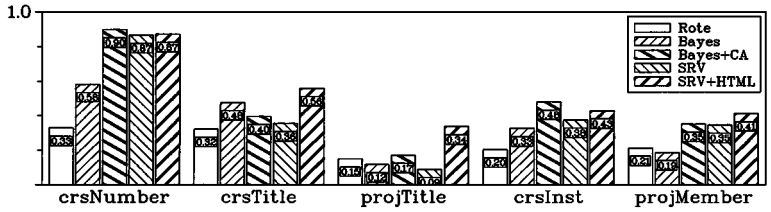

Figure 3. Peak F1 scores of the four learners on fields from the WebKB domains. SRV is tested both with the default featureset and with HTML-specific features.

values—they are optimized for recall. With several fields—notably etime—these learners issue many low confidence predictions and lose many precision points at the high-recall end of their precision/recall curve. The peak F1 score reflects the true power of these learners on such fields.

Two comments about these results are in order. The label "BAYES+CA" stands for BAYES augmented with the "canonical acceptor," the ALERGIA grammar prior to any state merging. Our experiments with grammatical inference, presented in Section 3.5, yielded the surprising result that the canonical acceptor outperformed any grammar produced as a result of state merging, when combined with BAYES. Therefore, we use this particular configuration when comparing with other learning approaches. Also, in figure 3, results are shown for a fifth learning approach, labeled "SRV+HTML". This stands for SRV run on a feature set augmented with HTML-specific features. Details are presented in Section 3.4.

Table 7 shows the precision and recall scores of all learners on all fields. These are full-recall results; the recall scores reported are the highest the respective learners were able to achieve. In contrast, Table 8 compares precision performance at a fixed recall level of 25%. In each case, the predicions of highest confidence are gathered, such that 25% recall is achieved, and the precision at this point is measured. Tables 7 and 8 are provided in lieu of precision-recall graphs; each column is equivalent to sampling precision levels at a single point on the recall axis of such a graph. Because the results in Table 8 involve a single statistic, we have included error margins at the 95% confidence level.

We will discuss these results in Section 3.6. Nevertheless, we can already draw one important conclusion: although performance varies from problem to problem—as does the best-to-worst learner ranking—SRV and BAYES+CA generally outperform the other two learners. As we will see in Section 4, however, this does not mean that the two lesser learners are useless; we will find that a combination of all learners is consistently better than any individual one.

3.4. Designing features for HTML

SRV's default feature set is not optimal for use with some information extraction tasks involving HTML, as is made clear by the last two rows in each of the above tables. Although SRV's performance using its default feature set can be competitive with the best of the other approaches, it benefits significantly from the introduction of HTML-specific features. This is perhaps not surprising. The default features, as shown in Table 5, reflect none of the

Table 7. Precision and recall of four learners on fields from three domains. BAYES+CA is BAYES augmented with the canonical acceptor using the *m*-estimate transduction method. Note that the fields "course instructor" and "project member" were evaluated using an instance/prediction-based metric, rather than a document-based metric.

	Prec	*Rec*	*Prec*	*Rec*	*Prec*	*Rec*	*Prec*	*Rec*	*Prec*	*Rec*
	speaker		location		stime		etime			
ROTE	55.1	6.8	89.5	58.1	73.7	73.4	37.4	71.6		
BAYES	34.5	25.6	57.3	58.8	98.2	98.2	49.5	95.7		
BAYES+CA	46.5	53.6	67.6	67.5	94.2	94.1	39.8	84.8		
SRV	54.4	58.4	74.5	70.1	98.6	98.4	67.3	92.6		
	acquired		purchaser		acqabr		dlramt		status	
ROTE	55.9	11.5	43.2	10.6	26.2	13.9	58.7	37.2	43.1	52.3
BAYES	22.2	21.6	37.6	41.3	24.1	33.0	50.3	47.8	32.0	42.1
BAYES+CA	27.0	27.3	35.9	39.5	30.7	42.1	31.1	64.6	36.7	48.3
SRV	40.7	39.4	45.2	48.6	32.9	44.9	48.1	67.0	39.0	45.0
	crsNumber		crsTitle		projTitle		crsInst		projMember	
ROTE	70.8	21.6	44.3	23.8	37.5	9.3	71.8	11.9	74.8	11.4
BAYES	56.4	59.7	38.0	44.4	10.6	12.9	8.7	48.3	15.4	19.9
BAYES+CA	85.2	90.3	31.2	36.4	14.0	17.0	7.7	59.6	23.7	38.4
SRV	84.1	87.3	33.0	34.6	7.7	9.3	20.6	53.2	26.2	35.2
SRV+HTML	84.0	89.0	45.7	50.0	26.3	31.4	21.6	55.9	30.0	41.1

HTML-specific structure that is immediately evident to the viewer of a Web page and that page designers rely on to facilitate presentation.

Table 9 shows the features that were added to the default feature set to generate the results reported in the rows labeled "SRV+HTML." Two basic types of simple feature were introduced. The *in* features return *true* for any token occurring within the scope of the corresponding tag. The *after* features return *true* only for the single token following the corresponding tag. Most of the simple features shown in Table 9 are derived directly from HTML tags in this way. In addition, a couple of simple features generalize more primitive features. The feature in_emphatic, for example, is a disjunction of in_i, in_em, in_b, and in_strong. Finally, several relational features (the features in bold face in the figure) were added which capture relations between tokens occurring together in HTML tables. (Recall that by *relational* feature we mean a function such as prev_token that maps a token to another token in the same document.)

3.5. Experiments with grammatical inference

The learner labeled "BAYES+CA" in the above tables represents BAYES augmented with a fragment grammar, as described in Section 2.3. As the label suggests, the particular grammar

Table 8. Precision of four learners at the approximate 25% recall level. Error margins represent the 95% confidence level. Missing values indicate a learner did not achieve the desired recall level.

	speaker	location	stime	etime	
ROTE	—	99.0 ± 1.2	78.2 ± 4.0	79.4 ± 5.7	
BAYES	36.1 ± 3.5	97.7 ± 1.7	100.0 ± 0.0	100.0 ± 0.0	
BAYES+CA	80.4 ± 4.3	99.3 ± 0.9	100.0 ± 0.0	100.0 ± 0.0	
SRV	79.9 ± 4.4	98.6 ± 1.3	100.0 ± 0.0	100.0 ± 0.0	
	acquired	purchaser	acqabr	dlramt	status
ROTE	—	—	—	73.4 ± 5.7	69.7 ± 4.5
BAYES	—	52.3 ± 3.8	40.2 ± 3.7	74.0 ± 5.8	51.0 ± 4.1
BAYES+CA	28.6 ± 2.5	55.4 ± 3.9	49.6 ± 4.2	78.6 ± 5.6	61.9 ± 4.4
SRV	53.3 ± 3.7	59.0 ± 4.0	53.7 ± 4.3	72.1 ± 5.9	70.0 ± 4.5
	crsNumber	crsTitle	projTitle	crsInst	projMember
ROTE	—	—	—	—	—
BAYES	85.5 ± 8.3	71.1 ± 10.2	—	53.3 ± 6.3	—
BAYES+CA	96.8 ± 4.4	68.4 ± 10.3	—	71.1 ± 6.6	55.7 ± 3.3
SRV	92.2 ± 6.6	43.5 ± 8.7	—	43.8 ± 5.7	36.8 ± 2.6
SRV+HTML	92.2 ± 6.6	96.4 ± 4.9	42.9 ± 9.2	63.4 ± 6.6	56.9 ± 3.3

Table 9. HTML features added to the core feature set. Features in bold face are relational.

in_title	in_a	in_h
in_h1	in_h2	in_h3
in_list	in_tt	in_table
in_b	in_i	in_font
in_center	in_strong	in_em
in_emphatic	after_br	after_hr
after_p	after_li	after_td
after_th	after_td_or_th	
table_next_col	**table_prev_col**	
table_next_row	**table_prev_row**	
table_row_header	**table_col_header**	

used was the "canonical acceptor," a grammar that accepts no example that is not in the training set. Examples, in this case, are fragments that have been generated by a transducer produced using the *m*-estimate method.

This particular configuration was determined to be the best by a set of experiments involving the seminar announcement fields in which we tried all combinations of several

transduction methods and several settings of ALERGIA's generalization parameter, α. Specifically, we tried five methods for generating transducers:

***M*-estimates** Use *m*-estimates, as described in Section 2.3, with *m* set to 3. The small value of *m* favors low-coverage tests and larger alphabets.

Information gain Use information gain, as described above. Like *m*-estimates, this prefers tests that favor field tokens over non-field tokens, but with a bias for high-coverage tests and larger alphabets.

Spread 5 Choose tests that spread field tokens as evenly as possible into five bins. As discussed in Section 2.3, this method does not consider non-field tokens.

Spread 10 Like Spread 5, but spread over 10 bins.

Spread 20 Like Spread 5, but spread over 20 bins.

For each transducer, we tried four settings of α: *canonical acceptor*, 0.9, 0.8, and 0.5. Note that the canonical acceptor is not equivalent to ROTE, since the transduction step abstracts over the raw field instances. The other settings correspond to increasingly aggressive settings of the generalization parameter; lower settings yield smaller, more general grammars.

Figure 4 shows the effect of various settings of α with transduction method fixed as *m*-estimates, and figure 5 shows the effect of various transduction methods using the canonical acceptor grammar. The evidence from these results argues in favor of our choice of *m*-estimate transduction, on the one hand, and using the canonical acceptor, on the other, as the best representative configuration of "BAYES+ALERGIA" for comparisons with other

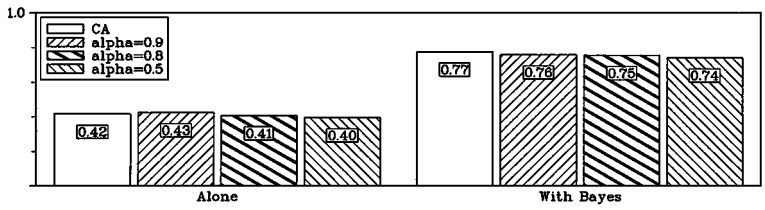

Figure 4. Average peak F1 score on the seminar announcement fields at various settings of ALERGIA's generalization parameter α. Both the performance of ALERGIA alone and BAYES augmented with ALERGIA is shown. The *m*-estimate transducer was used to generate all alphabets.

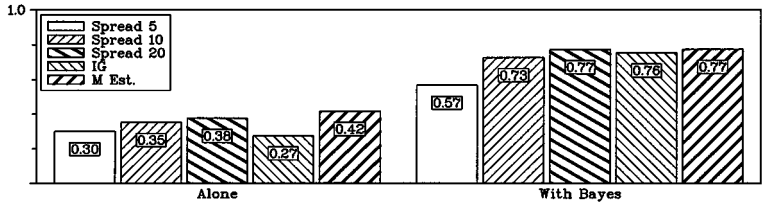

Figure 5. Average peak F1 score on the seminar announcement fields using various alphabet transduction methods. The canonical acceptor was used in all cases.

Table 10. Average size of decision lists generated using information gain and *m*-estimate metrics on the four seminar announcement fields.

	speaker	*location*	*stime*	*etime*
I. Gain	3.2	13.4	7.2	6.6
M. Est.	34.8	48.2	24.8	20.8

learning methods. It is interesting to note in passing that, although ALERGIA does poorly as a standalone predictor, its combination with BAYES yields one of the best learners we tested.

What are the factors that influence a good transduction for this problem? Is the size of the resulting alphabet important? Is it important to choose a transduction method that seeks to distinguish field tokens from non-field tokens? In conjunction with figure 5, Table 10 provides some insight. Alphabet (decision list) size is clearly a factor that influences the usefulness of the resulting grammar. This result ran counter to our expectations. We were concerned that large alphabets would stand in the way of effective generalization over the essential elements of field structure. Note that performance does not increase monotonically with alphabet size for all fields. Thus, it is not sufficient to have a large alphabet. However, it seems safe to say that, without a transduction method that optimizes for the main task (i.e., extraction), larger alphabet sizes increase the likelihood that important feature-value pairs are part of the transducer.

3.6. Discussion

In terms of both precision and recall, SRV tends to outperform the other learners tested here. SRV achieved the highest peak F1 score on 10 of the 14 fields. Part of SRV's power undoubtedly stems from its access to abstract features of tokens—access that two of the learners lack entirely, and that the third uses as part of an auxiliary task. Perhaps a fair comparison would somehow make these features available to ROTE and BAYES. It is not at all clear, however, how this might be managed without redesigning these two learners in a fundamental way. Moreover, as we argue in Section 4, by spending effort on optimizing individual learners we may be missing an opportunity to improve by combining learners.

SRV belongs to a family of algorithms—the *relational* family—that is very flexible with respect to the encoding of domain knowledge. A measure of its ability to exploit such knowledge is its performance on the WebKB fields. We note substantial performance improvements on most of these fields when HTML-specific features are provided to SRV. As discussed in Section 3.4, the features added for these particular runs were relatively obvious reflections of HTML structure. Implementing such a feature involves writing a function in C and compiling it into SRV.

The performance of BAYES+CA is comparable with, if usually lower than, that of SRV. The decomposition it embodies appears to be an effective one: BAYES, to locate approximately where field instances occur, and grammatical inference, to select fragments with the most appropriate structure. In a sense, BAYES+CA shows what happens when abstract

features are provided to BAYES. And inasmuch as we cannot claim to have discovered the best method for constructing the grammars used by BAYES+CA, it would not be surprising if an algorithm similar to BAYES+CA reached performance levels as high as SRV's.

Perhaps the most important conclusion is that there is no single best learning approach to these extraction tasks. On the status field, in fact, the simplest approach, ROTE, achieves the highest F1 score. It might be argued that this makes status less interesting as a test of learner generalization, but this would miss an important point: information extraction is not a problem susceptible to a single approach. Even a single field may comprise a number of sub-problems, each of which should be handled differently. Consider the performance of ROTE on the location field in Tables 7 and 8. Although ROTE does not achieve the recall levels reached by other learners on this field, there is some subset of location instances for which it is to be trusted over other learners. The question is how to know when to trust it.

4. A multistrategy approach

If forced at this point to choose a single learner from the four presented, we would recommend a relational approach like SRV. In light of the above considerations, however, a better recommendation might be "don't choose." Rather than use a single learner, would it not be better to use an automated approach for choosing among the learners for a given task, or even for combining their predictions in an attempt to achieve performance superior to any individual learner? Might we expect such a strategy to yield improved performance?

4.1. Opportunity

Either of two conditions might hamper or degrade our ability to realize combined performance better than that of the best individual learner:

1. The best learners may be close to a performance ceiling beyond which it is difficult to climb without a more sophisticated handling, such as one involving natural language understanding.
2. The best learners may simply subsume the others, predicting correctly on all document for which the lesser learners predict correctly.

If either of these conditions obtain, then probably the best we can do is to select the most consistent learner. If, on the other hand, individual learners solve different parts of the problem space, then there is hope for performance superior to that achieved by any single learner.

Table 11 suggests that such improvement is indeed possible. Taking the speaker field as a target, it shows a lack of agreement among learners on a surprisingly high fraction of documents. Performance numbers shown for SRV and BAYES+CA may leave the misleading impression that their document-by-document behavior is similar. This table shows that it clearly is not. In fact, of the 757 documents for which at least one of these learners predicted correctly, only 399 were handled correctly by both. If we possessed an oracle for this field which, in every case where the two learners disagreed, told us which of the two to

Table 11. Outcome contingency table for the speaker field. If R is the row learner and C is the column learner, then the entry in (R, C) is the probability that R handled a document correctly, *given* that C handled it correctly.

	ROTE	BAYES	BAYES+CA	SRV
ROTE	1	0.22	0.12	0.09
BAYES	0.81	1	0.40	0.30
BAYES+CA	0.93	0.85	1	0.66
SRV	0.81	0.68	0.72	1

trust, we could achieve 68% precision at 73% recall. Compare this with the performance of SRV, which manages 54% precision at 58% recall. And surprisingly, the oracular figure is not necessarily an upper bound. Both SRV and BAYES+CA make predictions which have confidence levels that are too low to figure in the results. In cases where top predictions from both learners disagree and are both wrong, they may be issuing predictions of lower confidence which agree on the correct fragment. Thus, although both individual learners may be wrong, it may be possible nevertheless to combine them for a correct prediction.

Here, we sketch a strategy for combining learners which we have given fuller treatment elsewhere (Freitag, 1998). Our idea is a particular variety of multistrategy learning which treats individual learners as black boxes and attempts to work only with their outputs. Our idea is to use the precision/recall behavior of a learner and statistical methods to build a model to map prediction confidence to probability of correctness.

4.2. Combining learners

A key assumption of our approach is that the probability that a learner's prediction is correct increases with increasing confidence. We can observe a decline in precision scores with increased recall by comparing Tables 8 and 7, suggesting that this assumption is at least partly warranted. Note that we do *not* assume that the confidence scores themselves bear any resemblance to true probabilities. The scores produced by BAYES, for example, are large negative log probabilities.

The idea, briefly, is this: *For each learner, use regression to model the relationship between confidence scores and probability of correctness.* Given such models, we can treat the predictions for a given fragment as probabilistic estimates that it is an instance. The specific steps involved in our approach are:

1. **Validate performance on a hold-out set.** Reserve a part of the training set for validation. After training each learner, store its predictions, with confidences, on the hold-out set.
2. **Use regression to map confidences to probabilities.** Based on the learner's performance on the hold-out set, attempt to model how its performance varies with confidence.
3. **Use the regression models and calculated probabilities to make the best choice on the test set.** Having used the validation set to calibrate learner confidence scores, we re-train all learners on the entire training set and use the regression models to combine their predictions on the test set.

4.3. Experiments

We present one combining strategy which has been consistently effective across all information extraction tasks with which we have experimented. In the modeling step we withhold one third of the training data and train all learners on the remaining two thirds. We then use linear regression to model the relationship between confidence and probability of correctness. For a given learner, the range between the lowest and highest confidence predictions seen on the validation set is divided into 10 intervals of equal size. Predictions are then distributed into 10 bins, depending on which interval they fall into. Once all predictions have been collected from a learner, we use the bins to generate ten datapoints (x, y) for regression, where x is the midpoint of an interval and y is the ratio in the corresponding bin of number of correct predictions to total number of predictions. Linear regression over these datapoints then completes the modeling step.

The number of bins is a parameter with which we briefly experimented in preliminary work—enough to determine that the quality of results does indeed depend on it.[4] The number 10 worked well on several fields, so we kept it fixed in later experiments. Ideally, this number would be set independently for each learner and field.

Given a field, the regression step yields a linear equation for each individual learner. The confidence of any prediction by a learner on a test fragment is now converted to a probability estimate by means of the respective equation. If a test fragment has drawn only a single learner's prediction, then the output confidence is simply this estimate. The interesting question is what to do with multiple predictions for a single fragment. Suppose P is the set of probabilities (mapped using the regression model from confidence scores associated with learner predictions) for a given fragment. Our combined estimate of the probability that the fragment is an instances is:

$$P_{\text{out}} = 1 - \prod_{p \in P} (1 - p) \tag{11}$$

This is the same as Eq. (9), which is used to combine the confidence scores of multiple matching rules in SRV. Given a document, the confidences derived in this way are used to re-sort all fragments extracted by any of the individual learners.

Table 12 shows the result, by means of peak F1 scores, of this procedure on the 14 fields. ROTE, BAYES, BAYES+CA, and SRV were all used as input to the combining method. For the WebKB fields we used the HTML-specific feature set in training SRV. Improvement over the best individual learner is seen for all but one field; in some cases improvement is substantial.

5. Related work

None of the individual learning methods described in this paper is completely new, either as a general machine learning approach or as applied to the information extraction problem. Several MUC researchers mention system components that have a form or function similar to one of the learners presented here.

Table 12. Peak F1 scores of the multistrategy approach compared with that of the best individual learner. Individual scores marked (⋆) belong to BAYES+CA; the score marked (⋆⋆) belongs to ROTE; all other individual scores belong to SRV.

	speaker	location	stime	etime	
Best individual	58.3	73.7	98.5	94.0	
Combined	66.2	79.7	99.3	94.3	
	acquired	purchaser	acqabr	dlramt	status
Best individual	40.7	47.9	39.6	60.8⋆	50.9⋆⋆
Combined	45.6	51.8	43.1	64.3⋆	58.6⋆⋆
	crsNumber	crsTitle	projTitle	crsInst	projMember
Best individual	89.9⋆	55.9	33.7	48.1⋆	41.1
Combined	88.9⋆	62.0	34.1	49.8⋆	45.5

An important component in many MUC systems is a domain-specific dictionary containing keywords and phrases peculiar to the target domain (Noah & Weeks, 1993; Muraki, Doi, & Ando, 1993). These dictionaries are typically constructed by hand. ROTE can be regarded as an automated method for constructing such a dictionary. To our knowledge, the performance of approaches like ROTE for information extraction has not been studied. ROTE, as a general machine learning approach, is a convenient baseline against which more sophisticated approaches can be compared (Mitchell, 1997). It can be regarded as a degenerate form of several learning approaches, such as k-nearest neighbor (Duda & Hart, 1973) or decision table learning (Kohavi, 1995).

Naive Bayes also can be regarded as a baseline method (Mitchell, 1997). Unlike ROTE, however, Naive Bayes can be competitive with more powerful methods (Domingos & Pazzani, 1996). The idea of a Bayesian method for text classification is almost as old as the idea of automatic text classification itself (Maron, 1961). Lewis discusses the uses of such methods for text classification and retrieval (1992) and provides a large list of pointers to related research (1997). In the MUC context, statistical methods have been used for various sub-tasks of the larger information extraction problem, including pre-extraction filtering (Cowie et al., 1993), syntax modeling (Weischedel et al., 1993), and slot filling (August & Dolan, 1992).

Beginning with Gold (1967), the problem of grammatical inference has both undergone considerable theoretical development and found diverse application (Vidal, 1994). The state-merging methodology, as exemplified by ALERGIA, is not the only viable approach to the problem. Error-correcting grammatical inference is one common alternative (Rulot & Vidal, 1988). Stochastic regular grammars are closely related to Markov models, a family of algorithms that appear to be suitable for use in information extraction (Rabiner & Juang, 1986).

Rivest introduces the notion of decision lists and discusses the class k-DL of decision lists having conjunctive clauses of size k at each decision. In these terms, the algorithm

described here produces lists belonging to 1-DL. In broader terms, the phrase "decision list" is frequently used to describe algorithms in the covering family, such as AQ (Michalski, 1983), CN2 (Clark & Boswell, 1991), and FOIL (Quinlan, 1990).

Finite state machines have been used to model linguistic syntax in at least one successful MUC system (Appelt et al., 1993), but they appear to have been manually designed. Kushmerick discusses learning patterns from a sub-class of the regular languages, specifically for the purpose of performing extraction from Internet documents (Kushmerick, 1997). In contrast with the experiments described here, his work on wrapper induction works directly with the source text and focuses on learning discriminative contextual patterns. Closer to the work presented here are systems described by Goan et al. (1996), and Bikel et al. (1997), each of which uses grammatical inference in conjunction with some kind of abstraction over textual units—over characters in the former, tokens in the latter—for the named entity extraction task. In the former, ALERGIA is modified to explore a pre-defined character-type hierarchy. In the latter, a static, manually defined decision list is used to abstract over tokens.

Rule-learning symbolic algorithms have been successfully applied to both the document classification problem (Apté, Damerau, & Weiss, 1994; Cohen & Singer, 1996) and NLP tasks related to information extraction (Cardie, Aone & Bennett, 1996; McCarthy & Lehnert, 1995). Early work in learning extraction patterns focused on the induction of simple rule-like features, which were then manually installed into larger information extraction systems (Riloff, 1996; Kim & Molfobsn, 1995).

CRYSTAL was the first system to treat the information extraction (i.e., "slot-filling") task as a supervised learning problem in its own right (Soderland, 1996). CRYSTAL is a covering algorithm, which conducts a specific-to-general (bottom-up) search for extraction rules. Rules in CRYSTAL are generalized sentence fragments. The feature space CRYSTAL searches is implicit. It consists of literal terms, syntactic relations, and semantic noun classes. A potential strength of the approach taken by CRYSTAL, compared with SRV, is its ability to generate rules to extract instances of multiple fields in concert. Each unique combination of fields encountered together in some sentence in the training data is treated as a separate learning problem.

RAPIER is closest to the approach described here (Califf, 1998). RAPIER is a relational learner designed to handle informal text, such as that found in Usenet job postings. Like CRYSTAL, RAPIER searches for rules bottom up. Unlike CRYSTAL, RAPIER searches the less structured space of unparsed text fragments. RAPIER is relational in that it in principle can exploit unbounded context surrounding field instances. It generalizes by dropping constraints from, or introducing disjunctions of constraints to, overly specific rules. Constraints are either actual terms, which must appear in the text for a rule to match, or part-of-speech labels. It appears, however, that RAPIER could be adapted to exploit the kind of typographic information used by SRV.

Of the work presented in this paper, the combination of BAYES with grammatical inference is most similar to what has traditionally been called *multistrategy learning* (Michalski & Tecuci, 1994). Originally, the term was most often used to refer to the combination of analytical and inductive methods, but the combination of inductive methods having different biases is consistent with the term "multistrategy." This has been called "empirical

multistrategy learning" (Domingos, 1996). We have also used the term "multistrategy learning" to refer to our framework in which individual learners are used as input to a combining method, which treats them as black boxes and attempts to model their behavior in pursuit of improved performance. Similar work has been called *meta-learning* (Chan & Stolfo, 1993). Cross-validation has been shown to be an effective way to *select* from among multiple candidate learning methods (Schaffer, 1993). In the cited study, however, a single learner is selected for an entire test set, and no attempt is made to combine learner predictions for individual examples.

6. Conclusion

It is possible to perform information extraction from informal text of the sort commonly found in the online environment, such as email, Usenet posts, and plan files, even in the absence of syntactic and semantic information. Machine learning is a rich source of ideas for algorithms that can be *trained* to perform this extraction. With the right machine learning techniques, very simple document representations make it possible to train effective extractors. Here, we have shown how to create such extractors using only a term-space representation and how to exploit typographic information in the form of simple token features.

Learning methods motivated by ideas in document classification, grammatical inference, and inductive logic programming, all have a part to play in this problem. However, in no case does any single algorithm subsume all the others. Rather, different algorithms solve different parts of the greater problem. This has led us to look for effective multistrategy approaches which can take the predictions of all available learners under advisement and produce extractors as good as or better than the best individual learner. We have described in outline one such approach.

However, we do not hold all pieces to this puzzle. A number of areas remain to be explored or fleshed out:

From field to record. We have treated each field in a domain as a separate task and target for learning. While this decomposition lends clarity to our study, it leaves open the question how single field results should be combined to produce a complete record—and how learning might exploit co-occurrence of fields in a document.

Loose multistrategy learning. We feel we have only begun to explore the potential of the ideas presented in the last section. This notion of *loose multistrategy learning*, in which the output behavior of learners is modeled and used to combine predictions, seems to hold promise for problems other than information extraction. And while it may be strictly less powerful than more traditional multistrategy approaches, it is flexible and modular. Both the details of how regression is to be conducted, as well as the right methods to use in combining predictions, need to be explored.

Grammatical inference. Our experiments suggest that much of the power of representing field structure lies in the transduction step. We believe that integrating this step with construction of the grammar would lead to an improvement in performance.

Layout. One source of information potentially useful for information extraction is document formatting. We have not yet attempted to exploit this source of information. Perhaps

ideas similar to those motivating our experiments with grammatical inference would prove useful.

Bayes. The way in which BAYES represents field instances is arbitrary. Rather than a fixed-size context window, it might be useful to have the context window size depend on the problem. Similarly, we are interested in ways to treat multi-term contextual fragments as features, rather than each term separately.

SRV. We continue to seek better ways to combine independently learned and validated rule sets. The work of other researchers interested in using covering algorithms for information extraction also suggests ways in which SRV might be extended.

The notion of information extraction is an important one for many applications on the research horizon. As the discipline broadens in scope to include problems that bear less and less resemblance to those explored in MUC, new sources of information and problem representations become important. Machine learning will enable us to construct systems which can quickly adapt to these changing conditions.

Acknowledgments

Thanks are due to Tom Mitchell, under whose advisorship this work was conducted, and to the other members of the WebKB and text learning groups at CMU for data and ideas. Thanks also to William Cohen and the anonymous reviewers for their meticulous comments and suggestions. This research was supported in part by the DARPA HPKB program under contract F30602-97-1-0215.

Notes

1. In general, documents from other domains may also be present in a collection, so that some sort of filtering must be performed either before or during extraction. For the purposes of this paper, we will consider this task ancillary to the problem of extraction and not discuss it further.
2. In keeping with common usage in relational learning, we call the individual antecedents in a rule *literals*. The space from which candidate literals are selected is defined by *predicates*, named templates that define the arity and semantics of literals.
3. Note that the requirement that predictions *exactly* match instances potentially discards many useful predictions. Indeed, some of the learners, particularly BAYES, see sizable jumps in both precision and recall when the criterion of correctness is that a prediction *overlaps* a field instance.
4. One might also treat the predictions themselves as datapoints. In our experiments, this resulted in performance inferior to that of the binning method.

References

Aone, C. & Bennett, S.W. (1996). Applying machine learning to anaphora resolution. In S. Wermter, E. Riloff, & G. Scheler (Eds.), *Connectionist, statistical and symbolic approaches to learning for natural language processing* (pp. 302–314). Berlin: Springer-Verlag.

Appelt, D.E., Hobbs, J.R., Bear, J., Israel, D., & Tyson, M. (1993). FASTUS: a finite-state processor for information extraction from real-world text. *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)* (pp. 1172–1178).

Apté, C., Damerau, F., & Weiss, S.M., (1994). Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, *12*(3), 233–251.

August, S.E. & Dolan, C.P. (1992). Hughes Research Laboratories: Description of the trainable text skimmer used for MUC-4. *Proceedings of the Fourth Message Understanding Conference (MUC-4)*, pp. 189–196.

Bikel, D.M., Miller, S., Schwartz, R., & Weischedel, R. (1997). Nymble: a high-performance learning name-finder. *Proceedings of the Fifth Conference on Applied Natural Language Processing* (pp. 194–201).

Califf, M.E. (1998). Relational learning techniques for natural language information extraction. Ph.D. Thesis, University of Texas at Austin.

Cardie, C. (1993). A case-based approach to knowledge acquisition for domain-specific sentence analysis. *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)* (pp. 798–803).

Cardie, C. (1997). Empirical methods in information extraction. *AI Magazine*, *18*(4), 65–79.

Carrasco, R.C. & Oncina J. (1994). Learning stochastic regular grammars by means of a state merging method. In R.C. Carrasco & J. Oncina (Eds.), *Grammatical inference and applications: Second international colloquium, ICGI-94*, Springer-Verlag.

Chan, P. & Stolfo, S. (1993). Experiments on multistrategy learning by meta-learning, *Proceedings of the Second International Conference on Information and Knowledge Management (CIKM 93)* (pp. 314–323).

Clark, P. & Boswell, R. (1991). Rule induction with CN2: some recent improvements. In Y. Kodratoff (Ed.), *Machine learning—EWSL-91* (pp.151–163). Springer-Verlag, Berlin.

Cohen, W.W. & Singer, Y. (1996). Context-sensitive learning methods for text categorization. *Proceedings of the 19th Annual International ACM Conference on Research and Development in Information Retrieval* (pp. 307–315) Zurich, Switzerland: ACM Press.

Cowie, J., Guthrie, L., Jin, W., Wang, R., Wakao, T., Pustejovsky, J., & Waterman, S. (1993). CRL/Brandeis: Description of the Diderot system as used for MUC-5. *Proceedings of the Fifth Message Understanding Conference (MUC-5)* (pp. 161–179).

Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., & Slattery, S. (1998). Learning to extract symbolic knowledge from the World Wide Web. *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*.

Defense Advanced Research Projects Agency (1992). *Proceedings of the Fourth Message Understanding Conference (MUC-4)*, McLean, Virginia. Morgan Kaufmann Publisher, Inc.

Defense Advanced Research Projects Agency (1993). *Proceedings of the Fifth Message Understanding Conference (MUC-5)*, Baltimore, Maryland. Morgan Kaufmann Publisher, Inc.

Defense Advanced Research Projects Agency (1995). *Proceedings of the Sixth Message Understanding Conference (MUC-6)*. Morgan Kaufmann Publisher, Inc.

Domingos, P. (1996). Unifying instance-based and rule-based induction. *Machine Learning*, *24*(2), 141–168.

Domingos, P. & Pazzani, M. (1996). Beyond independence: Conditions for the optimality of the simple Bayesian classifier. *Proceedings of the Thirteenth International Conference on Machine Learning (ICML-96)* pp. 105–112).

Doorenbos, R., Etzioni, O., & Weld, D.S. (1997). A scalable comparison-shopping agent for the world-wide web. *Proceedings of the First International Conference on Autonomous Agents*.

Duda, R. & Hart, P. (1973). *Pattern classification and scene analysis*. New York: John Wiley and Sons.

Freitag, D. (1998). Multistrategy learning for information extraction. *Proceedings of the Fifteenth International Machine Learning Conference (ICML-98)*.

Freitag, D. (1999). Machine learning for information extraction in informal domains. Ph.D. Thesis, Carnegie Melon University.

Goan, T., Benson, N., & Etzioni, O. (1996). A grammar inference algorithm for the World Wide Web. *Working Notes of the AAAI-96 Spring Symposium on Machine Learning in Information Access*.

Gold, E. (1967). Language identification in the limit. *Information and Control*, *10*, 447–474.

Kim, J.-T. & Moldovan, D. I. (1995). Acquisition of linguistic patterns for knowledge-based information extraction. *IEEE Transactions on Knowledge and Data Engineering*, *7*(5), 713–724.

Kohavi, R. (1995). The power of decision tables. *Proceedings of the European Conference on Machine Learning (ECML-95)* (pp. 174–89).

Kushmerick, N. (1997). Wrapper induction for information extraction. Ph.D. Thesis, University of Washington. Tech Report UW-CSE-97-11-04.

Lewis, D. (1992). Representation and learning in information retrieval. Ph.D. Thesis, Univ. of Massachusetts. CS Tech. Report 91–93.

Lewis, D.D. (1997). Reference list to accompany the SIGIR-97 Tutorial on Machine Learning for Information Retrieval. `http://www.research.att.com/ lewis/papers/lewis98.ps`.

Maron, M. (1961). Automatic indexing: An experimental inquiry, *Journal of the Association for Computing Machinery*, *8*, 404–417.

McCarthy, J.F. & Lehnert, W.G. (1995). Using decision trees for coreference resolution. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*.

Michalski, R. & Tecuci, G. (Eds.). (1994). *Machine learning: A multistrategy approach*. San Mateo, CA: Morgan Kaufmann.

Michalski, R. S. (1983). A theory and methodology of inductive learning. In R.S. Michalski, J.G. Carbonell, & T.M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (pp. 83–134), Palo Alto, Ca: Tioga Publishing Company.

Mitchell, T.M. (1997). *Machine learning*. The McGraw-Hill Companies, Inc.

Muraki, K., Doi, S., & Ando, S. (1993). NEC: Ddescription of the VENIEX system as used for MUC-5. *Proceedings of the Fifth Message Understanding Conference (MUC-5)* (pp. 147–159).

Noah, W.W. & Weeks, R.V. (1993). TRW: Description of the DEFT system as used for MUC-5. *Proceedings of the Fifth Message Understanding Conference (MUC-5)* (pp. 237–248).

Quinlan, J.R. (1990). Learning logical definitions from relations. *Machine Learning*, *5*(3), 239–266.

Quinlan, J.R. (1993). *C4.5: Programs for machine learning*. San Mateo, Calif: Morgan Kaufmann Publishers.

Rabiner, L.R., & Juang, B.H. (1986). An introduction to hidden Markov models. *IEEE ASSP Magazine*, *3*(1), 4–16.

Riloff, E. (1996). Automatically generating extraction patterns from untagged text. *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)* (pp. 1044–1049).

Riloff, E. & Lehnert, W. (1994). Information extraction as a basis for high-precision text classification. *ACM Transactions on Information Systems*, *12*(3), 296–333.

Rulot, H. & Vidal, E. (1988). An efficient algorithm for the inference of circuit-free automata. In G.A. Ferrate (Ed.), *Syntactic and structural pattern recognition*. Springer-Verlag, Berlin.

Schaffer, C. (1993). Selecting a classification method by cross-validation. *Machine Learning*, *13*(1), 135–143.

Soderland, S. (1996). Learning text analysis rules for domain-specific natural Language processing. Ph.D. Thesis, University of Massachusetts. CS Tech. Report 96-087.

Soderland, S. (1997). Learning to extract text-based information from the world wide web. *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*.

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

van Rijsbergen, C.J. (1979). *Information Retrieval*. Boston: Butterworths, Inc.

Vidal, E. (1994). Grammatical inference: an introductory survey. In R.C. Carrasco & J. Oncina (Eds.), *Grammatical Inference and Applications: Second International Colloquium, ICGI-94* (pp. 1–4) Springer-Verlag.

Weischedel, R., Ayuso, D., Boisen, S., Fox, H., Ingria, R., & Matsukawa, T., Papageorgiov, C., MacLaughlin, D., Kitagawa, M., Sakai, T., Abe, J., Hosihi, H., Miyamoto, Y., & Miller, S. (1993). BBN: Description of the PLUM system as used for MUC-5. *Proceedings of the Fifth Message Understanding Conference (MUC-5)* (pp. 93–107).

Received April 13, 1998
Accepted March 3, 1999
Final Manuscript March 3, 1999