Using Grammatical Inference to Improve Precision in Information Extraction

Dayne Freitag
School of Computer Science
Carnegie Mellon University
`dayne@cs.cmu.edu`

Abstract

The field of *information extraction* (IE) is concerned with applying natural language processing (NLP) and information retrieval (IR) techniques to the automatic extraction of essential details from text documents. We are exploring the use of machine learning methods for IE. While the most promising methods we have developed perform well for problems defined over a collection of electronic seminar announcements, they are imprecise in their identification of the boundaries of relevant text fragments (*fields*). Here, we entertain the idea of using grammatical inference (GI) methods to learn the appropriate form of a field. We describe one method for translating raw text into an abstract alphabet suitable for GI, and show that, by combining one IE learning method with the resulting inferred grammars, large improvements in precision can be realized for some fields.

Introduction

The recent explosion in the availability and accessibility of online text documents has given strong impetus to disciplines which seek to manage large document collections. The relatively well-explored fields of *information retrieval* and *information filtering* have finally come into their own as sources of useful tools and ideas for managing information glut. These fields aim to buffer the user at the document level, fetching or screening off whole documents based on some notion of the user's needs.

In contrast, *information extraction* (IE), a younger field of endeavor, attempts to summarize documents in terms of the essential details they contain, allowing the user to survey document collections more easily. In the simplest form of IE, a successful summary of a document is simply a collection of text fragments from the document, each fragment answering a specific question relevant to documents in the collection. For example, the experiments described in this paper involve a collection of seminar announcements taken from electronic bulletin boards in the CMU online environment. For each such document, the extraction problem, as we define it, is to identify the seminar location, starting time, ending time, and speaker.

From machine learning's point of view, information retrieval, filtering, and extraction, all can be regarded as classification problems. Whereas the goal of information retrieval and filtering is to classify an entire document as relevant or irrelevant with respect to a user's query (or profile, respectively), IE can be viewed as text fragment classification. Given a seminar announcement and my desire to identify the location of a seminar, I can ask a machine learner to classify fragments of contiguous text as `location` or `non-location` (i.e, as candidate location *fields*).

We have found this perspective useful in our experiments in the seminar announcement domain. Techniques shown to work for document classification also work, with suitable modifications, for simple information extraction tasks. The most common such techniques employ the "bag-of-words" document representation, in which documents are regarded as term vectors. Unfortunately, this representation discards structural information. Consequently, while a bag-of-words learner can be very effective at identifying the approximate location of relevant text fragments, it can have difficulty locating the correct field boundaries.

In reviewing some of the alignment problems experienced by the learner described in this paper, we were left with the impression that, in many cases, a simple notion of the syntactically legal form of a field (e.g, start times may have the form `[number, colon, number, number]`) would enable the learner to eliminate trailing irrelevant tokens from its predicted field, for example, or reject an inappropriate fragment as implausibly structured. To test this intuition, we used grammatical inference to capture field structure. We hypothesized that, armed with the probability that a novel token sequence has the appropriate syntactic structure for a field, the bag-of-words learner's precision would improve.

Information Extraction

Although the idea of automatically capturing the gist of text is not new, the field of information extraction is relatively young. It was conceived in the late 1980's out of a desire to put existing NLP technologies to

```
 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
```

Figure 1: A sample of the formatting styles used in seminar announcements.

practical use. In terms of linguistic sophistication, the quintessential information extraction task lies somewhere between information retrieval and full text understanding. Document collections used in standardized tests at the Message Understanding Conference (MUC 1992) (MUC 1993), the primary IE forum, consisted of newswire articles or technical texts. The traditional IE task, therefore, may require linguistic and domain knowledge, and typically necessitates some post-processing of raw extracted text fragments.

In our research, we are interested in extraction from the kind of text typical in online environments, such as email, netnews articles, and Web pages. Because such text is usually written to less formal standards than the kind of text considered at MUC, the nature of the extraction problem is somewhat different. The emphasis on grammatical prose and stylistic sophistication is replaced by devices that are more direct and concise. Because there is often a common means of delivery (e.g., a Web browser), which provides formatting guarantees, typical devices are often visual and two-dimensional, rather than linguistic and linear.

A good example of the kind of document collection that interests us is the data set used in these experiments. It consists of 485 electronic seminar announcements posted in the CMU environment between October, 1982, and August, 1995. These postings range considerably in style, from relatively formal visiting lecture announcements to information reminders of project group meetings. In general, all these postings share the purpose of announcing or reminding of an upcoming local gathering of people. The information extraction task, therefore, is to find the text fragments providing the details specific to such gatherings. It is not hard to argue for the usefulness of an extraction system in this domain. Such a system could be used, for example, in conjunction with an automated appointment manager, monitoring bulletin boards for seminars of interest to the user and presenting essential details for the user's perusal.

For the purposes of these experiments we defined four fields: the location of the seminar, its starting and ending times, and the name of the speaker. Figure 1 shows fragments from three documents in the collection, illustrating the variety of styles used to convey the details we want to extract. It is typical of the collection that only one of these three fragments is grammatically well-structured. The other two fragments illustrate two common devices, the one a terse, pointed listing of details with labels, the other a more visually oriented presentation. Note that not all fields can be instantiated for all documents. The first and third fragments do not contain ending times; the job of the extraction system in such a case is to indicate that the field does not occur.

Machine Learning and IE

The definition of the information extraction problem assumes the existence of a coherent domain of discourse to which documents in the collection belong. None of the documents in the seminar announcement domain, for example, was posted to advertise a computer for sale. This assumption of semantic coherence, along with the limited understanding required by the task definition (compared with story summary systems, for example), makes the NLP problem tractable and permits useful IE solutions, even in linguistically sophisticated collections.

A couple aspects of IE make it an attractive application for machine learning. For one thing, the problem is structured in a way very similar to the traditional ML problem: The document collection, or corpus, typically includes a tuning set. One designs one's IE system to do as well as possible on the tuning set in the hope that it will perform well when confronted with novel documents from the same semantic universe. This maps naturally to the traditional ML paradigm, with its training/testing distinction.

Another attractive aspect of the IE problem, from the perspective of ML, is its potential simplicity. Although general IE does not allow us to avoid the difficult problems of NLP—such as word sense disambiguation and discourse analysis—in many cases it is possible to solve the bulk of an IE task while skirting these issues. Several leading systems in the IE community, for instance, demonstrate that thorough syntactic parsing is mostly unnecessary, that useful extraction is frequently possible given cheap local syntactic analysis. (See, for example, (Appelt *et al.* 1993) and (Lehnert *et al.* 1992).) Of course, "simple" does not mean "easy" in this case; it only means that full text understanding, along with some of its more bedeviling questions, can be avoided in favor of more superficial methods in some domains. We gravitate to such domains, since they yield a clearer picture of the power and limitations of the ML techniques we are developing. As a consequence of this choice of focus, it is difficult to compare our techniques directly to systems described in the IE literature; MUC domains appear to require considerably more linguistic sophistication and knowledge-based processing than the domains we have experimented with. There is no scarcity of either kind of domain in the online environment, however, so we are confident that we have targeted an essential part of the general IE problem.

The intersection between machine learning and information extraction appears to be a fertile area for research. Many systems fielded for the MUC problems have included machine learning components for relatively well-tried sub-problems, such as syntactic tagging. Recently, however, several researchers have demonstrated the suitability of machine learning algorithms for purposes more closely tied to the problem of IE. The Hughes Trainable Text Skimmer, for example, used both k-nearest neighbor and Bayesian methods to estimate the likelihood that a text fragment instantiates a target field (August & Dolan 1992). The CIRCUS system showed that parse fragments could be used as the basis for learned extraction patterns (Lehnert *et al.* 1993). Soderland developed CRYSTAL, a trainable IE module, as part of the CIRCUS project, and explored the use of several traditional ML methods for IE (Soderland 1996). Riloff showed with the Autoslog system that useful extraction knowledge could be acquired without an annotated corpus, relying only on a relevant/irrelevant distinction between documents in the collection (Riloff & Shoen 1995).

While researchers in the IE community have turned their attention to ML methods, some machine learning researchers have shown an interest in the general IE problem. At the University of Washington, several projects have explored to possibility of machine learning for information extraction from World-Wide Web pages. The ShopBot, for example, is able to locate catalog listings for novel vendor sites on the Web (Doorenbos, Etzioni, & Weld 1996). Once there, it forms a model of the listing format, in order to extract details such as product name and price. In (Kushmerick, Weld, & Doorenbos 1997), this idea is elaborated into a general-purpose algorithm for inferring such patterns under certain assumptions. This can be regarded as special-purpose grammar inference for information extraction. The most pertinent (for this paper) result to come from this group, is the "Web Information Learner," an adaptation of the general GI algorithm Alergia targeted at the IE problem (Goan, Benson, & Etzioni 1996).

Naive Bayes

We have experimented with several machine learning techniques in the seminar announcement domain. One surprisingly successful technique is an adaptation of a fairly simple method from document classification, called *Naive Bayes*. The first step in classifying a document is to represent it in a form amenable to automatic methods, typically as a feature vector. The most widely employed representation is called *bag-of-words*, because under bag-of-words the individual elements in the document feature vector correspond to words that might occur in the document. The "bag" in "bag-of-words" presumably refers to the fact that all intra-document structure, linguistic or other, is discarded in example construction. If a word does not occur in a document, the corresponding element in bag-of-words representation of the document is typically zero. Otherwise, depending on the method, it is a boolean, reflecting the occurrence of the word; an integer word frequency; or a real-valued weight that indicates the importance of the word in the document. A very common method for setting weights is TFIDF (described, for example, in (Salton & Buckley 1987)).

Under a Bayesian approach to document classification, we consider each of the n possible classifications, $\{C_1 \ldots C_n\}$, of a document D as a competing hypothesis. Bayes rule,

$$\Pr(C_i|D) = \frac{\Pr(D|C_i)\Pr(C_i)}{\sum_{j=1}^{n}\Pr(D|C_j)\Pr(C_j)}$$

tells us how to combine our prior expectation of the document's classification (our estimate of each class's likelihood before we actually look at the document) with evidence taken from the document. Because we are only interested in classifying the document, and not in an estimate of actual likelihoods, we can disregard the denominator in the above expression, and simply assign document D to the class for which the product of our prior expectation ($\Pr(C_i)$) and our judgment based on document contents ($\Pr(D|C_i)$) is greatest.

Naive Bayes is "naive," because, in order to make the estimate $\Pr(D|C_i)$, it makes a great simplifying assumption: that each term occurrence in a document can be considered as evidence of class membership, *independent of any other term occurrence in the same document.* Formally:

$$\Pr(D|C_i) = \prod_{t \in D} \Pr(t|C_i)$$

where t is some word or term. Since we can estimate $\Pr(t|C_i)$ with a simple counting approach over the training set, this yields a decision procedure that is easy to implement and efficient to compute. (But see (Lewis & Gale 1994) for less "naive" approaches.)

Consider now the problem of identifying the location 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 location." (Call this hypothesis $H_{i,k}$.) In the case of a seminar announcement file, for example, $H_{309,5}$ might represent our expectation that a location field, say, consists of the 5 tokens starting with the 309th token. Note that, because we must concern ourself with identifying field boundaries correctly, a hypothesis requires two parameters for complete specification—the position of the hypothesized field and its length.

Given this formulation of the learning problem, a Bayesian treatment is straightforward. Our estimate of the prior probability of $H_{i,k}$ is simply our expectation, based on the training set, that a field would occur starting at the *ith* token and be k tokens long. In our implementation, we assume that these two probabilities, field position and field length, are independent, and we estimate them separately. Our estimate of the prior probability of a hypothesis, therefore, is:

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

In order to estimate the likelihood of the data given the hypothesis, $\Pr(D|H_{i,k})$, a straightforward adaptation of Naive Bayes would simply take the product of the individual term probabilities for the hypothesized field—again, something which can be readily estimated from the training data. Field lengths may vary considerably, however, and this approach has the unfortunate side-effect of handicapping longer fields. Consequently, in our implementation we take as our terms-in-field estimate the product of the mean individual term probability and the mean field length.

In contrast with document classification, an IE hypothesis involves identifying text fragments in the context of surrounding text. Thus, in forming our estimate of the likelihood of a field, we take into account a fixed-size window of tokens on either side of the hypothesized field. The intuition is that typical text will include clues to the occurrence of a nearby field. For example, a seminar location may be introduced with a "`Place:`" label. Furthermore, it is our hope that this additional context will help us not only find the approximate location of a field, but also hone in on its boundaries.

Training this learner simply involves taking a number of counts over the training set. In addition to taking statistics on the position and length of field, for each distinct token encountered in the training set, the system counts the number of times it occurs within the field of interest, as well as the number of times it occurs at each position within the fixed-sized window on either side of the field.

During testing, each possible field is examined, and the likehood that it is an instantiation of the desired field is esimated. Because, for a typical field, there is only a limited number of field lengths with non-zero probability, this basically involves a linear scan of a document. We take the field yielding the highest estimate as the system's prediction. In order to improve precision, we can also infer a threshold from the training data, and stipulate that the system fail to make a prediction below this threshold. Figure 2 shows an example field likelihood estimation.

Token	*Indiv. Prob.*	*Combined*	
`00`	-2.81	-9.23	Data Likelihood
`PM`	-2.12		
`Place`	-1.11		
`:`	-3.19		
Baker	-0.99	-3.77	
Hall	-0.91		
Adamson	-1.09		
Wing	-1.01		
`Host`	-1.75	-11.35	
`:`	-4.06		
`Hagen`	-2.77		
`Schempf`	-2.77		
Position		-4.99	Prior
Length		-2.84	
Posterior		-32.18	

Figure 2: A sample Naive Bayes field likelihood estimation for a location phrase ("**`Baker Hall Adamson Wing`**") taken from the seminar announcement collection. Tokens listed above the phrase in the *Token* column are those occurring before it in the text, while those below occur after. Scores are log probabilities. Note that the score for the "in-field" tokens is not the sum of their individual scores, but their average score multiplied by the mean length for location fields in this corpus. The final score is very high—essentially a "sure hit"—for this field.

Finding Field Boundaries

Although this method is surprisingly good at identifying field occurrences, it can have difficulty identifying precise field boundaries. Of course, this is not surprising, given the bag-of-words flavor of the learner. Although it does not employ a purely bag-of-words representation—the context windows provide ordering information—its lack of knowledge of field structure can cause it to fail to perform useful extraction, even when its sense of where a field occurs in a document is approximately correct. Often, an unintelligible piece of a field is extracted, or the extracted field contains tokens from the surrounding text which a human would consider trivial to filter out.

In general, poor alignment can be caused by a number of different things. As with Naive Bayes, an IE system may have insufficient grasp of the syntax and semantics of a field for reliable alignment. At the same time, even given the best possible system, it may not be possible to achieve perfect alignment. In one study involving a collection of technical texts on microelectronics from MUC, humans achieved only 82% preci-

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
```

Figure 3: Examples of poor alignment from actual tests of Naive Bayes.

sion at a 79% recall level (Will 1993). Thus, whether and where to label field boundaries can be open to considerable interpretation, and inconsistent labeling essentially guarantees some alignment error.

Naive 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 Naive Bayes's probability estimations 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. The fields defined for the seminar announcement domain provide a nice spread in these terms: Whereas almost all start times and end times can be constructed from a very limited token vocabulary, and tend to occur in contexts easily identified by the learner, locations and speaker fields are more problematic. The speaker field is particulary challenging for the learner, since, not only are names highly unpredictable, but the speaker of a seminar is frequently introduced without superficial clues, as part of a prose description of an upcoming talk, for example. Figure 3 lists some sample poorly aligned predictions from Naive Bayes's run through the data. Each such prediction is Naive Bayes's most highly rated fragment for some test file.

Using Grammatical Inference

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 Figure 3, even without a notion of the semantic content of tokens. 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 in instantiations of the speaker field. Similarly, without almost any exception, names do not contain numerals, while seminar locations usually do.

It is this hole in the learner's understanding that we hope to fill with grammatical inference. We hypothesize that even an overly general notion of appropriate field syntax will improve the system's precision when combined with Naive Bayes's understanding of field contents and context. To test this hypothesis, we selected two algorithms from the GI literature and trained them on field contents.

Pre-processing for GI

Before we can apply these algorithms, however, we must transform the data into a suitable format. The GI learners we tested expect a set of positive sequences consisting of symbols from a finite alphabet. Of course, the raw field instances already meet this criterion, since they are all sequences of ASCII characters, but there is good reason to believe that this is not the right abstraction level at which to apply GI. For one thing, the meaning of a field does not lie in individual characters, but in tokens or lexemes—conglomerations of characters. For another, the generalization of GI methods is driven by the I/O similarity between states in an automaton under construction; representing the problem in terms of characters reduces the opportunity for these methods to exploit higher-level regularities in the sequences.

Performing GI over the output of the tokenizer used by Naive Bayes, on the other hand, is a reasonable idea. In the case of highly stereotypic fields, such as start time and end time, this should in fact be sufficient, since the system rarely encounters tokens in these fields that it has not seen during training. As noted, however, these are not the fields with which Naive Bayes has much difficulty. Moreover, grammars learned over fields like speaker, where novel tokens are frequent, will have at least some of the same difficulty as Naive Bayes given raw tokens as input.

Instead, we require some abstraction over tokens in field instances, so that we can say something useful about the syntax of a hypothesized field even in the face of novel tokens. We can easily imagine what kinds of features might be useful for identifying the appropriate structure of a field, features such as `Capitalized?` and `Numeric?`. Our idea, given such a feature set, is to proceed as follows:

1. Tokenize field instances.
2. Translate tokens into abstract symbols.
3. Infer grammars over the abstract sequences.

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. In other words, we use a deterministic procedure to decide which feature is most salient for any given token. We will call this procedure an "alphabet transducer," for lack of a better term.

Inferring Alphabet Transducers

It is clear that no one token representation is equally appropriate for all fields. Imagine representing tokens in terms of the number of characters they contain, so that token sequences are represented as integer sequences for GI. Even this very general representation might permit, in the case of start and end times, the inference of grammars useful for this application, since times tend to be tokenized into unusually short tokens arranged in predictable patterns. For location and speaker, however, this approach would probably prove wholely inadequate, since they consist in large part of true words which vary in length. Generally, we do not want our alphabet transducer to abstract too aggressively from the token sequences. Consider, for example, the token `Dr`, a frequent component of the speaker field, as in `Dr. Jones`. If we uniformly replace speaker tokens with their capitalization profile (e.g., turning `Dr. Jones` into `[Cap, Dot, Cap]`), we run the risk of failing, at grammatical inference time, to distinguish between the uses of the period in the sequences `Dr. Jones` and `John K. Smith`. One period abbreviates an honorific, while the other punctuates a middle initial. Although failing to make this distinction may not be fatal, we might expect better precision from a grammar that can make it.

We want tokens like `Dr` in the speaker field, therefore, to pass through the transducer in more or less recognizable form, while other, less common tokens are replaced by some field-dependent abstract symbol. Rather than tinker with the feature set until, for each field, we came up with a transducer that seemed to work well, we developed an inference procedure for constructing a transducer similar in flavor to covering algorithms like CN2 (Clark & Niblett 1989) and Foil (Quinlan 1990).

The input to this procedure (Infer-Transducer) is two sets of tokens, Positive and Negative—all tokens occurring inside (and outside, respectively) the desired field—and a set of features defined over tokens. At each step during inference, Infer-Transducer picks one feature-value pair as the most salient trait of tokens in Positive, as defined against the background distribution represented by Negative. All tokens matching this pair are removed from the Positive set, and this selection step repeats. The output of this procedure is simply the list of feature-value pairs, in the order inferred. Figure 4 shows a sample of the procedure's output for the speaker field. We chose a simple measure of feature-value saliency: the probability that a token belongs to Positive, given that it matches the feature-value. Thus, that a token has the case-insensitive form `dr` yields the highest probability, in the training set that produced the list in Figure 4, that it belongs to a speaker field. In these experiments, we required that a

```
((word-lower "dr")
 (word-lower "mr")
 (word-lower "hancock")
 (word-lower "steve")
 (word "Smith")
 (word-lower "prof")
 (word "Cavalier")
 (word "Jeff")
 (word-lower "christel")
 (word "John")
 (capitalized-p t)
 (word ".")
 (sentence-punctuation-p t))
```

Figure 4: An alphabet transducer for the speaker field. The first item in each pair is a token feature, the second a possible value of that feature. During transduction, each token in a sequence is replaced by a symbol corresponding to the first pair it matches in this list. The `word` feature is simply the literal token, while `word-lower` converts all characters to lower case, thereby permitting a case-insensitive equality test over tokens.

feature-value pair match at least five tokens in Positive for it to be considered as an entry in the decision list.

During alphabet transduction, each token is matched individually against the learned decision list and replaced by a symbol corresponding to the first matching feature-value pair. If none of the tests matches the token, it is translated into the placeholder symbol `some-token`. For example, given the decision list in Figure 4, the field `Dr. Jones` would be translated into the three-symbol sequence `[word-lower+dr, word+., capitalized-p+t]`.

GI algorithms

For these experiments we applied two GI algorithms described in the literature, Alergia and ECGI. Alergia is an example of a state-merging approach to GI (Carraso & Oncina 1994). Given a set of sequences which are supposed to be drawn from some language, Alergia begins its inference with a maximally specific prefix-tree automaton which accepts all and only the sequences in the training set. Alergia generalizes by merging states in this automaton which it deems sufficiently similar. All state pairs are considered for merging, and the algorithm terminates once this (quadratic) pass over the automaton states is complete. States are considered similar if (1) they have similar acceptance behavior (accept the same fraction of the time); (2) they have similar transition behavior (transition on the same symbols with the same probabilities); and (3) the states to which they transition are similar enough for merging. The determination of similarity is statistical and is controlled by a parameter α, which ranges from 0 to 1. Given a low α, Alergia is more permissive in its state-similarity test, and aggressive merging and generalization result; a high α yields a more rigorous similarity test. In the experiments described here, α was set to 0.8. We used our own implementation of Alergia.

ECGI is an incremental approach to GI (Rulot & Vidal 1988). Beginning with a grammar G, which describes exactly one sequence in the training set, ECGI incrementally repairs G, so that it also accepts subsequent sequences, at each step making minimal changes to G. In determining how to do least damage to G, a distance measure is used to compare the new sequence with sequences encoded by G. A number of distance measures have been implemented for ECGI. In these experiments, we used the Ecgiq Toolkit, the implementation of ECGI by its originators. We trained ecgiq with the longest-common-subsequence distance metric (the default) and tested using the "forward" option.

Both methods are instrumented so that the grammars they infer return a probability of membership in the language they represent when presented with a novel sequence. This both makes it possible to treat an inferred grammar as an extraction function in its own right, and serves as convenient glue. In order to integrate Naive Bayes with Alergia, for example, we treated Alergia's membership estimate of a proposed field $Pr_A(D|H_{i,k})$ as independent of Naive Bayes's estimate $Pr_B(D|H_{i,k})$. Thus, we took the combined estimate to be simply the product of the two individual estimates:

$$Pr_{AB}(D|H_{i,k}) = Pr_A(D|H_{i,k}) Pr_B(D|H_{i,k})$$

In cases where a grammar failed to accept a sequence, we assigned the sequence a very small probability.[1] It was our hypothesis that this combined estimate would provide a more precise notion of field boundaries than is available to basic Naive Bayes.

Figure 5 shows an excerpt from a location automaton inferred by Alergia during one pass through the data. We extracted this partial automaton (the full automaton, trained on 324 location sequences, contained 100 states) by beginning with the initial state and repeatedly adding the state arrived at through the most probable transition. The figure shows all transitions between the states so collected. Emissions (in the inferred alphabet), along with their percentage transition probabilities, are shown next to arcs. Acceptance probabilities are shown next to accepting states (double circles). An example will hopefully elucidate the correspondence between raw field instances and automaton functionality. The automaton shown in Figure 5 would recognize the sequence `[Cap?, 'hall']` (which might correspond to the literal phrase "`Baker Hall`") and accept it with 6.9% probability. The assessed membership probability of this phrase would be $0.29 \cdot 0.295 \cdot 0.069$.

[1]Note that this is only necessary in the case of Alergia. ECGI by definition accepts all sequences with some probability.

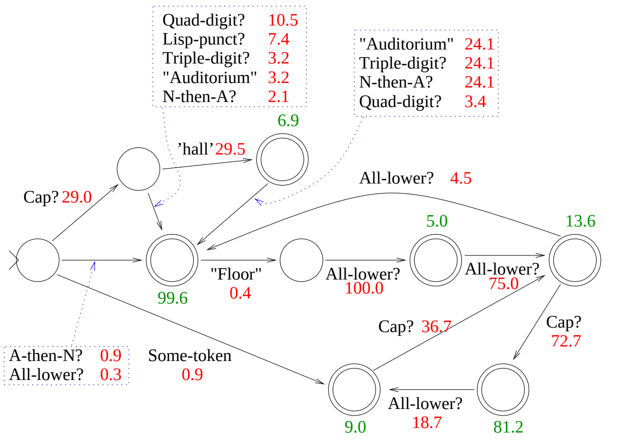

Figure 5: Parts of the Alergia automaton for location. Transition symbols in double quotes match field tokens in a case-sensitive manner, while single quotes indicate case insensitivity. Symbols without quotes correspond to more abstract token features.

Experiment

We divided the seminar announcement collection into training and testing sets ten times, yielding ten independent splits. For each split, each file in the collection was assigned with 50% probability to one of the two partitions, resulting in partitions of roughly equal size. Each learner was then trained on each of the training sets and tested on the respective test set. This occurred for each of the four defined fields.

All predictions made by the learners, along with their confidence, were collected for later analysis. The unit of measurement in the results reported here is a file. The scores reported here treat all test scores as if they came from one large test set. (There were 2397 test files, in this sense.) For each file, a learner's highest-confidence prediction was compared to the actual occurrences of a field. There are four possible outcomes from such a comparison:

Correct A field instance is correctly identified.

Wrong The best prediction is not a field instance.

Spurious The system makes a prediction for a file in which the field is not instantiated.

No prediction The system makes no prediction.

The measure of accuracy we report here is:

$$Accuracy = \frac{correct}{correct + wrong + spurious}$$

In other words, we are interested in the fraction of correct predictions over the number of all files for which the system makes some prediction.

Evaluation

It is a feature of the data and the problems defined over it that none of the learners encountered a file for which they could not make some prediction with non-zero confidence. Therefore, in order for a learner not to make a prediction on a particular file, it had to have a pre-set confidence threshold. Because start time is the only field instantiated in every file, and because our performance measure is sensitive to spurious predictions, some prediction threshold is desirable. It is harder to compare performance when all learners are forced to make a prediction for every file. This adds unnecessarily to the denominator of the accuracy measure and fails to highlight the strengths of a learner that is able to identify files not instantiating a field relative to a learner that is unable to do so.

Let *instantiated* be the number of files in the test set that actually contain the field of interest. We define the *coverage* of a learner's predictions as:

$$Coverage = \frac{correct + wrong}{instantiated}$$

which is simply the fraction of files having the field for which a learner has made some prediction. Because *spurious* predictions affect our accuracy measure, we desire learners that are adept at separating files with a field from those without a field. If a learner has this ability, then as its prediction threshold is raised, we can hope for significant gains in accuracy in exchange for somewhat reduced coverage.

Information retrieval and IE use two concepts closely related to accuracy and coverage: *precision* and *recall.* A number of techniques have evolved in these communities for measuring the extent to which a learner is able to trade coverage for accuracy. However, rather than apply one of these techniques, which can be difficult to interpret, we opt for the simpler solution of measuring accuracy at an arbitrary recall level. All performance numbers we report are approximately at the 80% recall level. To achieve this recall level, we set the prediction threshold manually.[2] Because setting a precise recall level can be difficult (or even impossible), we settled for figures close to 80%. Actual coverages are listed with accuracy in the reported results.

We have deferred until the now the question what constitutes a "correct" prediction. Rather than adopt a single criterion and risk either losing valuable information about learner performance or making the prediction task too simple, we measured each learner's performance under three criteria:

Overlap An extraction is correct if the predicted field overlaps an actual field. Note that this criterion does not guarantee useful behavior. It only gives a loose indication of the learner's ability to identify approximately the position of a field.

[2]Automatic methods for threshold determination are on our agenda for future research.

	Start Time		End Time	
Learner	*Cov.*	*Acc.*	*Cov.*	*Acc.*
Bayes	80.6	99.8	80.3	99.4
Alergia	81.2	47.3	81.8	32.5
ECGI	81.6	32.3	83.0	36.1
B+A	80.7	99.8	80.5	97.9
B+E	80.2	99.9	80.2	98.3

Table 1: Start time and end time performance results, *exact* criterion.

Contain The predicted field strictly contains an actual field, and at most k neighboring tokens. For the results reported here, $k = 5$.

Exact The predicted field matches exactly an actual field.

Note that these criteria can be ordered in terms of containment.

$$overlap \supseteq contain \supseteq exact$$

Consequently, a learner's performance cannot increase as we change criteria in the order listed. The degree to which a learner's performance decreases tells us how much difficulty it has locating field boundaries.

Results

The results for start time and end time presented in Table 1 demonstrate how easy these fields are for the Bayesian learner. Because the tokens occurring in these fields come from such a small set, and because their distribution differs so markedly from the general token distribution, Naive Bayes has little difficulty identifying these fields. Note that these scores use the "exact" criterion. Some of the strength of this learner must be due to its attention to context. The two fields were both chosen for this problem, because they are so easy to confuse when not considered in context. The GI learners' relatively poor performance undoubtedly can be attributed in part to this. That they do as well as they do indicates that there is some syntactic difference between the two fields. Indeed, certain times are more typically start times than end times; the inferred alphabet transducers help distinguish between the two.

In such a case, where a simple Naive Bayesian learner is able to solve the problem, we want the addition of grammatical information not to hurt performance. Therefore, the last two lines in Table 1 are gratifying. There is a little evidence that adding either Alergia or ECGI hurts Naive Bayes in the case of end time, but the damage is minimal and unlikely to be noticed in most applications.

The result for location, in Table 2, present a somewhat more interesting picture. The performance of Naive Bayes, which is quite strong, falls steadily as we change criteria in the direction of greater strictness

Learner	*Coverage*	*Overlap*	*Contain*	*Exact*
Bayes	81.0	97.8	81.5	72.0
Alergia	81.8	55.7	48.4	48.3
ECGI	81.2	57.4	15.2	15.2
B+A	80.3	97.4	79.7	79.0
B+E	80.1	98.9	75.4	73.8

Table 2: Location performance results.

Learner	*Coverage*	*Overlap*	*Contain*	*Exact*
Bayes	81.0	65.6	54.9	40.7
Alergia	80.2	7.0	4.6	4.6
ECGI	80.1	6.8	4.5	4.5
B+A	80.1	78.4	66.8	62.9
B+E	80.3	76.9	64.5	59.4

Table 3: Speaker performance results.

(from left to right). Thus, while it is nearly perfect at identifying the approximate position of a location field in a document, in a large number of cases (the difference between "overlap" and "contain") it fails to include part of the field, and in a substantial number of cases (the difference between "contain" and "exact") it includes junk tokens from neighboring text.

The most gratifying result, given our hypothesis, is the *exact* measurement for Naive Bayes + Alergia. The 79% score represents a significant improvement over the 72% achieved by Naive Bayes alone. Two other aspects of N+A's score are interesting. First, it roughly tracks the performance of Naive Bayes, so that including it does not hurt performance much under any criterion. Second, there is little change in performance between *contain* and *exact*. This is precisely what we would like to see: If the learner is able to find the whole field, its knowledge of syntax helps it draw precise boundaries.

The other salient aspect of this table are the low scores achieved by ECGI. Moving from *overlap* to *contain* and *exact*, yields a large drop in performance. We have not investigated this phenomenon, but it is clear that ECGI is somehow preferring field fragments over complete fields in some cases. It may be the case that, given its bias of minimal changes to a grammar under construction, ECGI is causing sequences to shares states in a way unnatural to the domain. Two common location phrases, for example, are "Adamson Wing, Baker Hall," and "Wean Hall 5409." If the word "Hall" in these two phrases is mapped to the same state in the inferred grammar, ECGI might prefer "Wean Hall" to "Wean Hall 5409," because of the abundance of location phrases that end with "Hall." Note that Alergia's merging heuristics do not suffer this problem, given enough data.

The results in Table 3 provide the greatest evidence in favor of our hypothesis. Not only does the integration of either GI method improve the precision of Naive Bayes's predictions, it improves the *overlap* accuracy as well. In other words, the addition of structural knowledge allows Naive Bayes to locate speaker fields where it failed without such knowledge previously.

Note that the large improvements in accuracy—about 20% under the exact criterion—come despite the fact that Alergia and ECGI both do quite poorly identifying speaker fields by themselves. This is an intriguing phenomenon. The two GI methods do poorly on speaker presumably because the alphabet transduction was more abstract on this field than on the others. The lack of frequent tokens forced alphabet inference to use an abstract feature such as `Capitalized?`. Nevertheless, Naive Bayes saw the greatest benefit from grammatical knowledge on the speaker field. It is worth investigating to what extent the two results—the more abstract alphabet and the substantial benefit from GI—are correlated.

Conclusion

There is clear evidence that, for some fields, using traditional grammatical inference methods in conjunction with another learning method provides real benefit to the information extraction task. Not only can precision in field boundary indentification be improved, but gains in overall accuracy can also be realized. We described one method for adding syntactic knowledge—the output of grammatical inference—to a Bayesian learner for information extraction.

Whereas grammatical inference has already been applied to the problem of textual structures with highly regular syntax, we outlined a pre-processing method that allows its application to general text. We sketched a learning technique for inferring what we have called "alphabet transducers," components which re-express text in a more abstract form that is conducive to GI treatment. We consider this a very promising idea for future research; we have not explored with any thoroughness the effect the form of the learned transducers has on the usefulness of inferred grammars.

It appears in general that grammatical inference can play a very fruitful role in information extraction, not as a solution by itself, but as a source of valuable structural knowledge for higher-level components in an integrated extraction system. We have only scratched the surface, however. Grammatical inference has been used to understand document layout, for example, something of potentially great benefit for information extraction (Ahonen & Mannila 1994). We might also ask what would happen if textual sequences were expressed in terms that reflected linguistic information. Would this enable the application of machine learning approaches, which have been shown to be successful in a relatively simple domain, to more difficult problems characterized by sophisticated linguistic structure? Finally, the transduction step appears to us in some sense superfluous and unnecessarily heuristic. It is interesting to ask how GI methods can be instrumented to obviate this step, taking as input feature vectors, rather than symbols, and determining feature salience as part of grammatical inference.

Acknowledgements Thanks are due to Tom Mitchell for repeatedly galvanizing the author's research into this problem with novel ideas and useful insights. Pierre Dupont helped broaden the author's understanding of grammatical inference in general, and ECGI in particular.

This research was sponsored in part by ARPA under grant number F33615-93-1-1330.

References

Ahonen, H., and Mannila, H. 1994. Forming grammars for structured documents: an application of grammatical inference. In Carrasco, R. C., and Oncina, J., eds., *Grammatical Inference and Applications: Second International Colloquium, ICGI-94*, 153–167. Springer-Verlag.

Appelt, D. E.; Hobbs, J. R.; Bear, J.; Israel, D.; and Tyson, M. 1993. FASTUS: a finite-state processor for information extraction from real-world text. *Proceedings of IJCAI-93.*

August, S. E., and Dolan, C. P. 1992. Hughes Research Laboratories: description of the trainable text skimmer used for MUC-4. In *Proceedings of the Fourth Message Understanding Conference (MUC-4).*

Carraso, R. C., and Oncina, J. 1994. Learning stochastic regular grammars by means of a state merging method. In Carrasco, R. C., and Oncina, J., eds., *Grammatical Inference and Applications: Second International Colloquium, ICGI-94.* Springer-Verlag.

Clark, P., and Niblett, T. 1989. The CN2 induction algorithm. *Machine Learning* 3(4):261–263.

Doorenbos, R. B.; Etzioni, O.; and Weld, D. S. 1996. A scalable comparison-shopping agent for the world-wide web. Technical Report 96-01-03, Department of Computer Science, University of Washington.

Goan, T.; Benson, N.; and 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.*

Kushmerick, N.; Weld, D.; and Doorenbos, B. 1997. Wrapper induction for information extraction. In *Submitted to the 17th International Joint Conference on Artificial Intelligence.*

Lehnert, W.; Cardie, C.; Fisher, D.; McCarthy, J.; Riloff, E.; and Soderland, S. 1992. University of Massachusetts: description of the CIRCUS system as used for MUC-4. In *Proceedings of the Fourth Message Understanding Conference (MUC-4).*

Lehnert, W.; McCarthy, J.; Soderland, S.; Riloff, E.; Cardie, C.; Peterson, J.; and Feng, F. 1993. UMass/Hughes: description of the CIRCUS system used for MUC-5. In *Proceedings of the Fifth Message Understanding Conference (MUC-5).*

Lewis, D. D., and Gale, W. A. 1994. A sequential algorithm for training text classifiers. In *Proceedings of the 17th International Conference on Research and Development in Information Retrieval.*

1992. *Proceedings of the Fourth Message Understanding Conference (MUC-4)*, McLean, Virginia: Morgan Kaufmann Publisher, Inc., San Francisco.

1993. *Proceedings of the Fifth Message Understanding Conference (MUC-5)*, Baltimore, Maryland: Morgan Kaufmann Publisher, Inc., San Francisco.

Quinlan, J. R. 1990. Learning logical definitions from relations. *Machine Learning.*

Riloff, E., and Shoen, J. 1995. Automatically acquiring conceptual patterns without an annotated corpus. *Proceedings of the Third Workshop on Very Large Corpora.*

Rulot, H., and Vidal, E. 1988. An efficient algorithm for the inference of circuit-free automata. In Ferrate, G. A., ed., *Syntactic and Structural Pattern Recognition.* Berlin: Springer-Verlag.

Salton, G., and Buckley, C. 1987. Term weighting approaches in automatic text retrieval. Technical Report 87-881, Department of Computer Science, Cornell University.

Soderland, S. 1996. *Learning Text Analysis Rules for Domain-specific Natural Language Processing.* Ph.D. Dissertation, University of Massachusetts. Avalable as Department of Computer Science Technical Report 96-087.

Will, C. A. 1993. Comparing human and machine performance for natural language information extraction: results for English microelectronics from the MUC-5 evaluation. In *Proceedings of the Fifth Message Understanding Conference (MUC-5).*