Machine Learning for Information Extraction from Online Documents
A Preliminary Experiment

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

1 Introduction

The experiment described here was designed for two things: to test the feasibility of a learning approach to information extraction in a real-world domain, and to uncover evidence that by using multiple learners it is possible to achieve better performance than by using a single learner.

Because the documents used in this experiment are taken unmodified from a real online environment designed for human-to-human communication, the task is a challenging one. Its difficulty varies considerably from field to field, but in all cases, in order to conclude that this approach is feasible, I require of each learner that its performance is substantially better than that of a random guesser. Of course, in practice the required performance level is defined by the intended application. Consequently, my argument for feasibility is informal. Some applications may be able to exploit a well-behaved precision-recall curve, so I look for this from the learners tested here.

We cannot expect to benefit by using multiple learners if all the learners score the same successes and make the same mistakes. Thus, a necessary condition of the multiple-learner scheme is that they differ in terms of behavior. I look for differences in two places:

1. The relative strengths of the learners should vary with the field under consideration, depending on how the field's profile interacts with a learner's bias.

2. Given a field, the less overlap between learners in absolute terms—whether a learner succeeds or fails on a particular file—the better.

Note that this evidence is *necessary* but not *sufficient*. This experiment does not explore ways to integrate learner predictions.

Although the learners' biases vary, they all share the basic approach. All are implemented using an internal function, $\mathcal{G}(D, f)$, as described in [1]. Any differences in behavior, therefore, are largely due to differences in bias and view.

Of course, this experiment addresses only a couple core claims of this thesis and leaves many of the conjectures and ideas raised here untouched. In partic-

Number of files:	485
Average size in bytes:	1890
Average number of tokens:	240
Average size of token in characters:	4.8
Average number of sentences:	7.6
Minimum number of sentences:	0
Maximum number of sentences:	177

Table 1: General document collection statistics.

ular, only three types of learner, of the four types listed, are tested, and only two document views are used, the terms view and the syntactic view.

2 Domain

The data set used in these experiments consists of 485 electronic seminar announcements posted locally between October, 1982, and August, 1995. The announcements in question are largely, but not exclusively, for seminars and meetings in the School of Computer Science and School of Robotics at Carnegie Mellon University. In addition to such announcements, there are also announcements for meetings at the University of Pittsburgh, announcements for political rallies, for self-help seminars, for club gatherings, etc. These postings range in style from relatively formal visiting lecture announcements to informal reminders of project group meetings. In general, all these postings share the purpose of announcing or reminding of an upcoming local gathering of people.

	Location	Start Time	End Time	Speaker
Total occurrences	648	987	437	761
Distinct phrases	233	122	80	486
Number of files with	464	485	228	409
Occurrences in sentences	132	122	30	247
Average per file	1.3	2.0	0.9	1.6
Minimum in file	0	1	0	0
Maximum in file	4	4	3	9

Table 2: Field statistics.

Of the many possible fields in this data set, I defined four for extraction: location, start time, end time, and speaker. Occurrences of these fields were identified in all documents and manually tagged. In addition, sentences were

```
The University of Pittsburgh-Carnegie Mellon Women's Speaker Series.

September 30th at 4:30 p.m.,CMU Faculty Dining Room in Skibo.
Panel Discussion:  ~Thriving in the Academy~
```

```
THURSDAY, MAY 18th
"Sonic Hedgehog and the Establishment of Embryonic Pattern", Dr.
Clifford J. Tabin, Department of Genetics, Harvard Medical School, 3:30
p.m., 1295 Biomedical Science Tower, Pitt.
```

```
DATE   ROOM      TIME   SPEAKER               TOPIC

10/28d HH B131   2:30   Dr. John Bares      Dante II and Beyond:
                                  CMU Field Robotics Exploration Robots
```

Figure 1: Some examples illustrating unconventional grammar and use of non-grammatical features.

tagged in all documents. Tables 1 and 2 give some of statistics of the resulting domain. Note in particular the low number of occurrences of the end time field, and the high number of occurrences of the speaker field, in sentences.

One thing becomes clear in setting up this experiment, which is perhaps not so clear when considering the problem of extraction abstractly: Extra-grammatical textual structures and ungrammatical sentences are very common. Rules of spelling, punctuation, and style, are applied creatively rather than followed. If they stand in the way of conveying essential details, they are frequently disregarded in favor of more visual techniques. The likelihood that a field appears in a parsable sentence varies considerably with the field type. These factors appear to argue strongly for a flexible approach involving multiple learners, rather than one involving intensive NLP. Figure 1 presents a sample from the collection of such ungrammatical structures.

3 Learners

For this experiment I implemented and tested three learners, which I describe in order from least to most sophisticated.

3.1 Dictionary learner

The dictionary learner (`Dict`) acquires and indexes all instances of a field in the training set. For each such entry `Dict` keeps two counts: the number of times it appears in the training set as an instantiation of the desired field (pos), and the number of times it appears in the rest of the text (neg). Given a test file, `Dict` scans for sequences of tokens[1] which, modulo whitespace, occur in its index. The probability that any such token sequence is an instance of the desired field is estimated as $pos/(1 + pos + neg)$. `Dict` chooses the highest-probability token sequence as its prediction, and always makes a prediction for a file if it finds a sequence that occurs in its index.

3.2 Vector-space learner

`Vect` takes a bag-of-words view of the problem, disregarding information about token order within field instances. On the other hand, it attempts to integrate information about tokens immediately preceding and succeeding fields observed in the training set. For each distinct token ($token$) in the training set, `Vect` keeps eight counts: the number of times $token$ occurs in any context ($total(token)$), the number of times $token$ occurs in a field instance ($in(token)$), the number of times $token$ occurs in each of the three token positions preceding a field instance ($before(token, 3)$, $before(token, 2)$, and $before(token, 3)$), and similarly for the three positions following a field instance ($after(token, 1)$, $after(token, 2)$, and $after(token, 3)$). In addition, `Vect` counts the number of times each distinct field length occurs in the training set. (Call this statistic $lengths$; $lengths(3) = 15$ means `Vect` saw 15 field instances containing exactly 3 tokens.)

Prediction involves combining all these frequencies into a confidence score in the range $[0, 1]$. `Vect` scans a test file for possible field instances, at each token boundary estimating the likelihood that it begins an instance of the desired field. This is done by hypothesizing a field of length k for each k such that $lengths(k) > 0$. For each such hypothesized field, `Vect` uses all its frequency information from the training set to produce a confidence score that the hypothesized field is indeed a field instance.[2] The highest such score is taken as the likelihood that a field instance begins at the token boundary under consideration. `Vect` records all fields with a score greater than 0.5.[3] `Vect`'s prediction is then the recorded field with the highest score. If no field is given a score higher

[1] I use the word "tokens" in describing the learners, rather than "words" or "terms," because not just alphabetic character sequences, but also units of punctuation are visible to them.

[2] Note that because `Vect` makes dubious independence assumptions in computing this score, it cannot be taken as a probability. Nevertheless, these assumptions introduce systematic errors, so that the score is useful for comparing the confidences of two competing field predictions.

[3] This value was determined to yield reasonable performance on the location task and subsequently used for all other tasks. Clearly, there are better, if slower, ways to find a good threshold value.

```
Function FieldAt(position)
  fieldtotal = total number of occurrences of field
  bestprob = 0
  FOREACH length such that lengths(length) > 0
    lenprob = lengths(length) / fieldtotal
    fieldprob = lenprob * ContentsProb(position, length)
    IF fieldprob > bestprob THEN
      bestprob = fieldprob
      bestlen = length
  RETURN (bestprob, bestlen) if bestprob > 0.5

Function ContentsProb(position, length)
  notprob = 1
  FOR i = 1 TO 3
    token = TokenAt(position - i)
    toktot = total(token)
    notprob = notprob * (toktot - before(token, i)) / (1 + toktot)

    token = TokenAt(position + length + i)
    toktot = total(token)
    notprob = notprob * (toktot - after(token, i)) / (1 + toktot)
  FOR i = 1 to length
    token = TokenAt(position + i - 1)
    toktot = total(token)
    notprob = notprob * (toktot - in(token)) / (1 + toktot)
  RETURN 1 - notprob
```

Figure 2: Pseudocode for `Vect`. The function `FieldAt` returns a confidence score and length for a proposed field starting at the given position. The `lengths` array holds the occurrence counts for fields of various lengths. The arrays, `before`, `after`, `in`, and `total`, hold occurrence counts for tokens in various positions relative to the field in the training set.

than 0.5, `Vect` declines to predict. This algorithm is summarized in figure 2.

3.3 Relational learner

The relational learner (`Rule`) takes a syntactic view of the extraction problem. Training sentences are parsed for syntactic information, then decomposed into a set of training examples, each example corresponding to a token. Thus, the training set for `Rule` consists of all tokens in parsable sentences in the set of training files. Associated with each such example are features which reflect the structure assigned to it by the parser, as well as simple contiguity information and orthographic details, such as capitalization. Many of the features, particularly those that reflect syntactic structure and contiguity, associate tokens in typed relations with other tokens in the same sentence. These are exploited by `Rule` to generate novel features during learning which capture multiple-token context.

I used Sleator and Temperley's link grammar parser as the source of `Rule`'s syntactic features.[2] Rather than tag each word in a sentence with its syntactic role, this parser binds words pairwise in typed relations ("links") which reflect their syntactic relationships. The link vocabulary—the set of possible links along with their annotations—is quite rich and able to capture a wide range of syntactic structure. More important for the purpose of this experiment, the relational form of the parser output lends itself to easy adaptation for use with

```
  +------------------------------Xp-----------------------------+
  +----Wd----+                                                  |
  |    +--G--+              +-----Os-----+     +------Js-----+  |
  |    +X+   +--Ss--+---I---+      +--Ds-+--Mp-+   +--G-+--G-+  |
  |    | |   |      |       |      |     |     |   |    |    |  |
///// Dr . Jones will.v address.v the group.n in Wean Hall 5409 .
```

Figure 3: A link-grammar parse of the sentence, "Dr. Jones will address the group in Wean Hall 5409.

a relational learner, such as `Rule`. An example parse is shown in figure 3; the bibliography entry for [2] includes a pointer to a document detailing the meaning of individual links.

For a given field type, `Rule` learns three sets of rules, one for each of: the left field boundary, the right field boundary, and the main token in the field. So that `Rule` could learn to identify the left field boundary, for example, all tokens in the training set which occurred as the first in a field of the desired type were marked as positive examples, all other tokens as negative examples. The "main" token was identified heuristically by assuming the field in question to be a noun phrase and following links to the main noun.

Given a particular sub-concept, `Rule` induces a ruleset in the spirit of Foil and CN2. Like Foil, `Rule` is allowed to add to its feature set during inference by deriving features based on inter-token relations. We constrained `Rule` to perform this feature derivation only when a base relational feature was chosen as a predicate in the rule under construction. When this happens, a set of features is added which are based on the selected base relational feature. For example, one relational feature in the feature set provided to `Rule` is `previous-word`, which returns the predecessor of a word in the sentence. If `Rule` selects `(previous-word = "of")` as a predicate, then before continuing with inference `Rule` generates a set of features derived from `previous-word`, such as `(previous-word is a noun)` and `(previous-word of previous-word)`. In this way, given enough data `Rule` can create features that capture context involving the features of many words.

After induction is finished and the rules have been pruned on the training set, rather than throw away any rules, `Rule` records the accuracy of each rule on the training set. This triple of rulesets, one for each of the field elements listed above, along with statistics measuring individual rule performance on the training set, is then returned by `Rule` as the learned extractor for the field. Figure 4 shows some sample rules from one run, as well as the sentences containing the tokens they match.

During testing, predictions produced by each of these rule sets must be assessed for accuracy and somehow combined to produce a predicted field instance

```
IF (previous-word does not exist) AND
   (a right G-link exists) AND
   (a right-G-token has a left W-link) AND
   NOT (a left G-link exists)
THEN this is a left boundary token for speaker.

Mr. Kurtz is a new genre artist and a founding member of...


IF (previous-word does not exist) AND
   (a right G-link exists) AND
   (the next-word of the right-G-token is "will")
THEN this is a left boundary token for speaker.

In this presentation, Dr. Curlee will give an overview of...


IF (word is "Dr")
THEN this is a left boundary token for speaker.

Dr. Watanabe is a Member of Sony Corporation's Board of Directors...
```

Figure 4: Sample learned rules, along with sentences they match from the data set, for the left boundary token of the speaker field. The predicate, `(a right-G-token has a left W-link)`, is an example of a dynamically derived feature.

	Training		Testing	
	Have Field	Field in Sentence	Have Field	Field in Sentence
Location	95.3	19.4	95.8	19.4
Start Time	100.0	17.1	100.0	16.7
End Time	48.7	3.5	45.3	2.6
Speaker	83.8	29.2	84.9	29.3

Table 3: Fields visible to `Rule`. Percentage of files having a field and percentage of files with the same field in parsable text.

along with a confidence score. A rule set matches a token if at least one of its constituent rules matches. Of course, the rule that matches may be one of the high-coverage, low-accuracy rules generated in the latter stages of induction. Thus, we estimate the strength of the match using the statistics taken on the training set. The statistics from all matching rules in the set are combined, under an (again dubious) independence assumption, to produce a confidence score. If this score is above some low threshold, we consider the rule set to have fired in response to the token.

Testing occurs at the sentence level. All the tokens in a test sentence are shown to all rule sets. If all three rule sets fire in response to tokens which occur in the proper order and in close proximity, then we have no difficulty reconstructing the field that is presumed to be present. In this case, if a score combining the individual rule set scores (again assuming independence) is above a certain threshold, we return the field and the score. Some simple preliminary experiments convinced me, however, that requiring all three sets to fire is too restrictive. Consequently, I consider the possibility that a field instance is present if only two of three rule sets fire. In this case, if the failing rule set is for a field boundary, `Rule` heuristically finds the missing boundary and supplies a low-confidence quasi-prediction, using its score in the same way it would that of a real prediction.

4 Experiment

Ten random splits of the 485 seminar announcements were made. Each split was generated by assigning each document to the train or test set with equal probability, resulting in sets of roughly equal size. Each learner was then trained on each train set and tested on the corresponding test set. Learner predictions on the test files were recorded.

`Dict` and `Vect` were instrumented to work directly with training and test files. Because `Rule` was designed only to work with parsed text, however, it saw only a portion of the available field instances in training sets and could make

	Location	Start Time	End Time	Speaker
`Dict`	88.3 / 98.4	79.2 / 81.0	38.6 / 39.6	54.2 / 55.7
`Vect`	76.4 / 90.5	84.2 / 99.6	63.9 / 65.5	45.1 / 82.9
`Rule`	55.4 / 77.1	18.5 / 63.1	0.0 / 40.0	57.2 / 60.9
`Rand`	1.9 / 6.3	3.0 / 7.9	2.2 / 6.0	2.4 / 5.8

Table 4: Accuracy: Percentage of file predictions which correctly identified a field instance. Scores are given for both strict and lenient criteria.

predictions for only a portion of the test files.[4] Table 3 gives an idea of the relative amount of data available to `Rule`.

5 Results

The results of these runs are presented in Table 4. As noted above, all learners can opt not to give a prediction for any given file. The scores in Table 4 represent the number of correct predictions over the number of all predictions made by the learner. In all cases (as will be shown below) it is possible to trade coverage for accuracy. I made no attempt to do so in generating these scores; instead, I used "reasonable" default prediction thresholds chosen in initial tests.

What constitutes a "correct" prediction depends ultimately on the uses to which predictions are to be put in applications. Is it sufficient, for example, for a prediction simply to overlap a field instance?[5] Or must it align exactly with actual field boundaries? I adopted a criterion somewhere between these two extremes: A prediction was counted correct if it contained the actual field completely plus at most k neighboring tokens ($k = 5$ in these experiments). I call this the *strict* criterion. It is also interesting to ask what fraction of predictions are in the right ballpark, so I calculated a second score which gives the fraction of predictions that shared at least one token with a field instance—the *lenient* criterion.

To provide a rough idea of the difficulty of the task, I also calculated the expected performance of a random guesser (`Rand`). This guesser gets to consult an oracle that tells it whether or not a field instance is present in a test file and the length of the field. It then is allowed to set its predicted field to the optimal length for performance under the strict criterion (5 plus the length of the field). Thus, `Rand` has perfect coverage, and its accuracy is measured only on fields

[4] This is somewhat arbitrary. There is no reason why a learner of this sort can't be given non-syntactic features for unparsable text. On the other hand, this approach gives a better idea of the contribution syntactic information can make.

[5] Note that lax criteria like this one, unless qualified, lend themselves to a trivial solution: always predict a field that spans the entire document.

	Location	Start Time	End Time	Speaker
`Dict`	65.7	99.2	96.2	16.2
`Vect`	91.4	100.0	80.1	56.8
`Rule`	14.4 / 60.7	18.9 / 70.6	3.4 / 41.3	34.5 / 76.5

Table 5: Coverage: Percentage of files containing field for which predictions were made. The second number in the `Rule` row is the percentage of files for which *parsed sentences containing the field* were available.

that actually contain instances of the respective field.

Table 5 gives coverage figures for the learners on the individual tasks. These are simply the percentage of files containing the target field for which a learner made a prediction. Because the input to `Rule` consists only of parsable text, it actually saw fewer field instances than the other learners, in both the training and testing sets. How many fewer depended on the field type. The second number in `Rule`'s row, therefore, shows `Rule`'s coverage for test files in which the target field was visible to it.

A quick inspection of Table 4 shows that there are clear winners for the location and start time tasks. Therefore, in asking whether improvement is possible by combining learners, I focus on the end time and speaker tasks. We can expect little improvement if a single learner achieves high accuracy with high coverage.

Figures 5 and 6 present the accuracy/coverage graphs on these two tasks.[6] All predictions made by a learner were sorted by confidence score, from highest to lowest. The point x on the coverage axis corresponds to a learner's having made predictions on a portion of all files containing the target field equal to x. At that point, the y value means that of *all* predictions made—including files that don't contain the target field—the learner has predicted correctly for a fraction equal to y. Curves are shown for both the strict and lenient criteria.

Finally, Tables 6 and 7 present contingency tables for these two tasks. In each table the two learners are compared which performed best on the task. Each value gives the percentage of all test files for which a particular event occurred. For example, by scanning the *Correct* row in Table 6, we see that for about 30% of the test files both `Vect` and `Dict` made correct predictions, for about 3% `Vect` predicted correctly but `Dict` predicted incorrectly, in no case where `Vect` predicted correctly did `Dict` make a spurious prediction, and in 0.5% of the cases `Vect` predicted correctly but `Dict` made no prediction. *Wrong* means a file contained the target field, but a learner predicted some other field; *Spurious* means a file did not contain the target field, but a prediction was made,

[6] I say "accuracy/coverage" in preference to "precision/recall," since the latter has a specific technical meaning in IR which doesn't fit our problem perfectly.

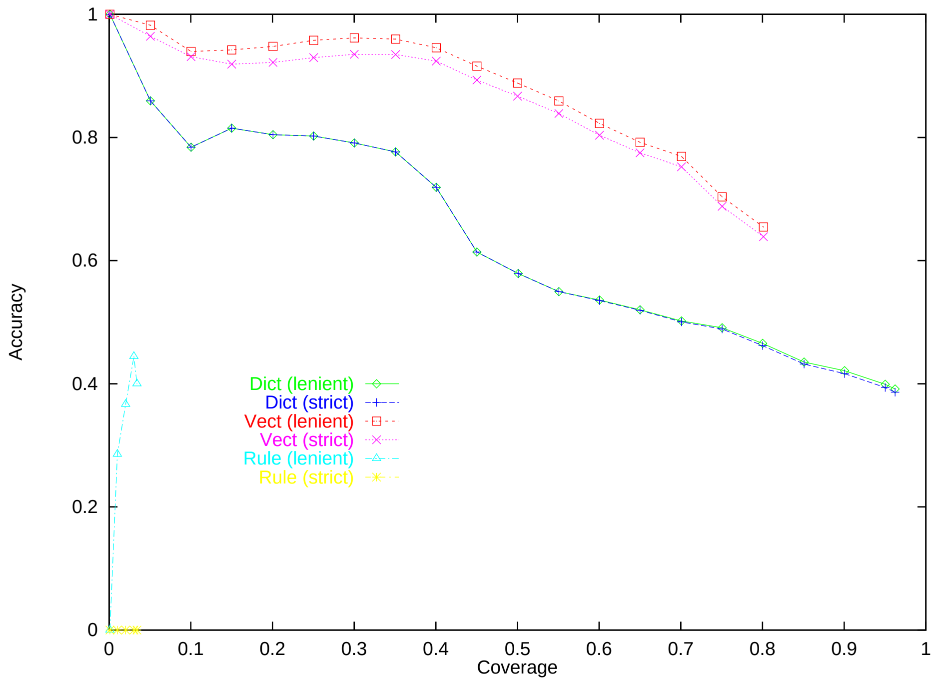

Figure 5: End Time: Coverage vs. Accuracy

		Dict			
		Correct	Wrong	Spurious	No prediction
Stat	Correct	29.9	3.3	0.0	0.5
	Wrong	0.8	1.8	0.0	0.0
	Spurious	0.0	0.0	0.0	0.0
	No prediction	4.3	3.5	47.1	8.3

Table 6: Contingency table for `Vect` and `Dict` on the end time task

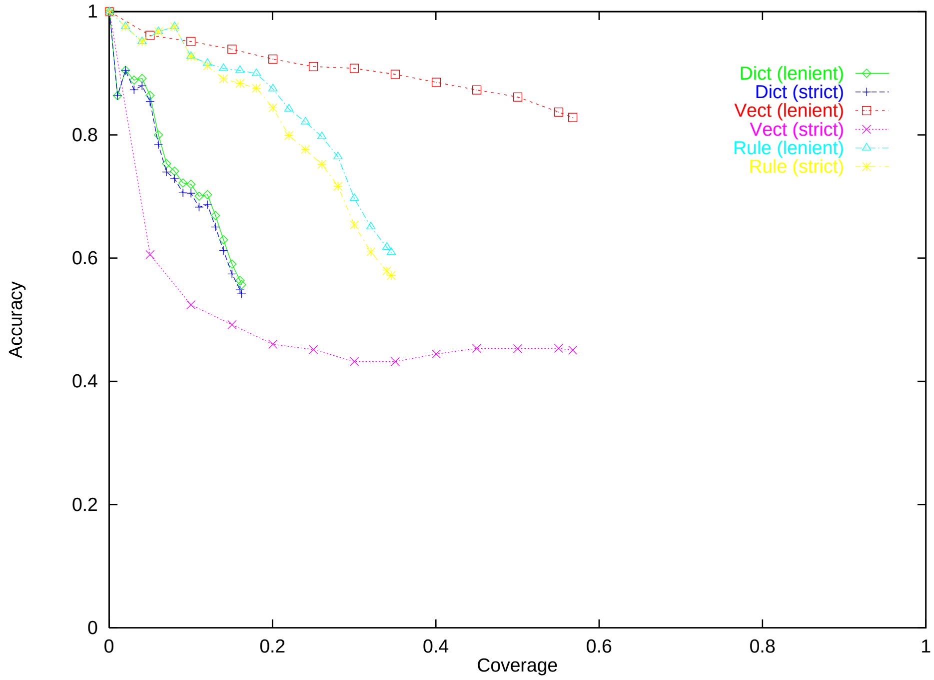

Figure 6: Speaker: Coverage vs. Accuracy

		Rule			
		Correct	Wrong	Spurious	No prediction
Stat	Correct	4.5	3.0	0.0	15.1
	Wrong	6.9	3.0	0.0	15.6
	Spurious	0.0	0.0	0.0	2.1
	No prediction	7.6	4.3	0.0	37.8

Table 7: Contingency table for `Vect` and `Rule` on the speaker task

anyway. All scores are computed using the strict criterion.

6 Discussion

The results give reason to believe that a machine learning approach to information extraction is indeed feasible, and that by combining multiple sources of information and multiple learners, better performance can be achieved than through a single learner. Below, I discuss these two points separately, then comment on the soundness of these conclusions, concentrating particularly on questions this experiment raises but does not answer.

6.1 Feasibility

While `Rand` is admittedly a strawman, giving no indication of the "true" difficulty of the individual extraction tasks, its performance nevertheless shows the size of the space an automatic learner must search to find a target field. Thus, there can be no doubt that each variety of learner is acquiring and applying useful extraction knowledge. With one exception, all learners outperform `Rand` by large margins on all tasks. While we might question the generality of the result, at least in this domain—which is real-world and motivated by an actual application—machine learning for information extraction yields useful results.

On some tasks, in fact, the performance of individual learners is so high that we can conceive of applying them immediately with little modification. The stand-out example of this is `Vect` on the start time task, where it achieves an accuracy of 99.6%, under the lenient criterion, while covering all test examples. Thus, while it may have a little difficulty identifying field boundaries as implemented, we can expect this problem to be mostly solved with a little additional research. Note that the start time field occurs in *all* files tested. Thus, we would be justified in considering the problem of identifying seminar start times from bboard announcements essentially solved.

6.2 Combining multiple learners

The strongest argument for using multiple learners is provided by differing field characteristics. The results make this clear. `Dict` performs especially well on the location task, where there is a relatively small number of possible values, and where a location phrase almost never can be confused with something else. (It is hard to imagine "Wean Hall 5409" standing for anything but a location.) By the same token, `Dict` does poorly on the speaker task, because in the CMU environment there are a large number of possible values for this field, and they are are seldom repeated. `Rule`, on the other hand, does reasonably well on the speaker task, since the speaker is the most likely field, of those presented here, to occur in grammatical sentences.

In fact, I selected the end time and speaker fields for this experiment, because they showcase the strengths and weaknesses of individual learners. `Dict` is able to identify start times reasonably well, because, again, times are hard to confuse with text in general. Because it ignores context, however, it has no way of distinguishing between start times and end times, insofar as they take on the same values. Consequently, it does poorly on end time, a task which requires some attention to context. Similarly, the speaker task lends itself partially to a statistical treatment that takes context into account. But to do well on it requires some knowledge of grammar.

Ideally, we could combine the strengths of two learners that do well on the speaker task, `Vect` and `Rule`, to yield performance that improves on either of them, in terms of both accuracy and coverage. While it is not clear how to do this, a prerequisite is that their behavior differ sufficiently that one learner can be used to cover the weaknesses of the other. Table 7 provides clear evidence that this prerequisite is met for this task. For example, if we knew to trust `Rule` in the cases where it predicts correctly but `Vect` predicts incorrectly (6.9%) or makes no prediction (7.6%), and if we could suppress prediction in the 15.6% of cases where `Vect` predicts incorrectly but `Rule` does not predict, we might realize accuracy gains of over 20%. In practice, of course, we probably cannot expect such improvement, but a significant increase in accuracy is by no means far-fetched.

6.3 Observations

The domain we chose for this experiment is atypical in traditional IE. As noted in the introductory discussion, IE typically concerns itself with collections of documents containing prose written to a high standard, such as newswire articles, the aim being an exploration of NLP techniques in a practical problem. By turning our attention to a domain where such prose is not the rule, we open the door for more statistically motivated techniques. The results for `Vect` reflect this. This is also reflected in `Rule`'s relatively low coverage and poorer performance on some fields.

In a sense, then, the extraction task is easier in this domain than in a traditional one. Nevertheless, I believe this domain is typical of a large number of domains available on the Internet and in the common online environment. The seminar announcement problem was motivated by the requirements of a real application. In order for this application to possess a desired functionality, the extraction problem had to be solved. One advantage of this particular take on information extraction, therefore, is its solid grounding in real-world needs.

The strength of `Vect` across all tasks is a surprise. Figure 6 points to some difficulty identifying field boundaries. This problem aside—looking at lenient scores—`Vect` performs better than any other learner on all tasks. At this stage, it is hard to say how useful `Vect` will prove for extraction problems in general. Because it has no access to syntactic or semantic information, we can expect its power to be limited to domains where stereotypic wording around and in field instances is the rule, such as this one. As noted above, however, this domain is typical of many others where extraction would be useful. The performance of `Vect`, therefore, holds the exciting promise of lightweight, high-accuracy extraction systems. Moreover, there is no reason to suppose that the performance of an algorithm like `Vect` would not improve with the addition of features capturing syntactic and semantic information. This latter point is an appealing direction for future research.

The most sophisticated of the learners, `Rule`, which represents a novel approach to the problem of field extraction, also appears to hold promise. Its relatively worse performance can be attributed partly to the much smaller number of training and testing examples it saw than the other learners, and partly to the large number of untested heuristics it incorporates. Clearly, its concept language enables it to express useful extraction patterns. Less clear is how much of a performance improvement might be achieved through improved heuristics, a richer concept language, and improved coverage of training and testing files.

Because of the differences in bias, we probably can never expect any one learner to solve the extraction problem outright or consistently outperform the others. The set of learners we have adopted for this experiment, however, seems to constitute an appropriate mix for the problem: We have a learner to cull off the very simple fields and to provide evidence of the existence of a field in cases where a field takes on a small number of possible values; another learner makes statistical judgments based on a small local context and appears to have wide applicability; and a third learner tries to make up for the system's lack of understanding by learning syntactic patterns for those fields that are too difficult for the other two learners. If this combination of learners does not solve the extraction problem to human standards, it at least seems to cover the problem well with existing techniques.

References

[1] Dayne Freitag. Machine learning for information extraction from online documents, 1996.

[2] Daniel Sleator and Davy Temperley. Parsing english with a link grammar. *Third International Workshop on Parsing Technologies*, 1993. A comprehensive description of link types and meanings is available at ftp://ftp.cs.cmu.edu/user/sleator/link-grammar/system-2.1/guide-to-links.