Sources of Success for Information Extraction Methods

David Kauchak
Dept. of Computer Science
UC San Diego
La Jolla, CA 92093-0114
dkauchak@cs.ucsd.edu

Joseph Smarr
Symbolic Systems Program
Stanford University
Stanford, CA 94305
jsmarr@stanford.edu

Charles Elkan
Dept. of Computer Science
UC San Diego
La Jolla, CA 92093-0114
elkan@cs.ucsd.edu

Abstract

In this paper, we examine Boosted Wrapper Induction (BWI) as an exemplar of recent rule-based information extraction techniques by conducting experiments on a wider variety of tasks than has previously been studied, including several natural text document collections. We provide a systematic analysis of how each of BWI's algorithmic components, particularly boosting, contributes to its performance over comparable methods. We show that the benefit of boosting comes from the ability to reweight examples to learn specific rules (resulting in high precision) combined with the ability to continue learning rules after all positive examples have been covered (resulting in high recall). We also propose the SWI-Ratio as a quantitative measure of the regularity of an extraction task, and show that this ratio is a strong indicator of IE performance. Based on these results, we present an overview of the current successes and limitations of rule-based IE systems as a whole. Specifically, we address limitations in the sources of information made available to IE methods, the current representations used by these system, and the relationship between confidence values returned during extraction and true probabilities. In this analysis, we investigate including grammatical and semantic information for natural text documents, as well as parse tree and attribute-value pair information for XML and HTML documents. We show experimentally that incorporating even limited grammatical information can improve both the regularity and performance of natural text extraction tasks. We conclude with novel suggestions for enriching the representational power of rule-based IE methods to exploit these and other types of regularities.

1 Introduction

Freely available text is abundant. Thousands of new web pages appear everyday. News, magazine, and journal articles are constantly being created. E-mail has become one of the most popular ways of communicating. All these trends result in an enormous amount of available text in digital form. This repository of text is an untapped resource of information. However, identifying specific desired information is not always an easy task. Information extraction (IE) is the task of extracting relevant fragments of text from larger documents to be processed later in some automated way such as responding to a user query. Examples of IE tasks include identifying the speaker featured in a talk announcement, finding proteins referenced in a biomedical journal article, and extracting the list of credit cards accepted by a restaurant from an online review.

A variety of systems and techniques have been developed to address the information extraction problem. Many successful techniques have included statistical models such as n-gram models, Hidden Markov Models and probabilistic context free grammars (Califf, 1998). Recently, though, rule-based systems that employ some form of machine learning have become increasingly popular and successful. These systems have taken a variety of different approaches, but have all recognized a number of common key facts. First, creating rules by hand is extremely difficult and time consuming (Riloff, 1996). For this reason, most of the systems generate the rules given raw unlabeled data or partially labeled data. Second, people have recognized that trying to generate a single, general rule for extracting a given field is often impossible (Muslea, et. al., 1999). Instead, most of the systems attempt to learn a number of rules that cover the training set and then combine these rules in some way.

One recent technique for generating rules in the realm of text extraction is wrapper induction. Wrapper induction techniques have proved to be fairly successful for IE tasks in highly structured domains, such as web pages generated from a template script

(Muslea, et, al., 1999; Kushmerick, 2000). However, because of their specificity, these methods do not generalize well to more natural texts, thus limiting their applicability.

Recent research in improving weak classification rules using boosting (Shapire, 1999) has led to a method for increasing the coverage of a weak learner by repeatedly learning rules that focus on portions of the decision space that previous rules have found difficult. Boosting works by continually reweighting the training examples, and using the weak learner to learn a new rule each time, stopping after a fixed number of iterations. This collection of rules is then combined by a weighted vote (related to their individual performance). Boosting has been shown theoretically to perform well and performs well in practice.

Boosted Wrapper Induction (BWI) is an IE technique that uses AdaBoost to generate a more general extraction procedure from a set of specific wrappers (Freitag, et. al., 2000). BWI has been shown to do well on a wide variety of tasks with partially structured and highly structured documents, but specifically how boosting contributes to this performance increase has not been investigated. Furthermore, BWI has been proposed as a potential solution for natural text, but little has been done to examine its performance in this challenging domain.

In this paper, we investigate the benefit of boosting in BWI and also its performance on natural text. We do this by comparing BWI's use of boosting with wrapper induction against another simple and common approach to combining weak learners, sequential covering. With sequential covering, the rules are ordered in some way according to "quality." The best rule is chosen, and all of the examples that the rule correctly classifies are removed from the training set. The process is then repeated until the entire training set has been covered. For a more detailed description of sequential covering see (Cardie, 2001). Sequential covering has been used in a number of systems (Califf, 1998; Clark, et. al., 1989; Michalski, 1980; Muslea, et. al., 1999; Quinlan, 1990) because it is fairly simple to implement, tends to generate understandable and intuitive rules and has achieved good results.

This paper is broken down into a number of sections. In section 2, we briefly describe BWI and related techniques, providing a formalization of the problem, along with a review of relevant terminology. In section 3, we present experimental results comparing these different rule-based IE methods on a wide variety of document collections. In section 4, we analyze the results of these experiments in more detail, with specific emphasis on how boosting affects BWI's performance, and how performance relates to the regularity of the extraction task. In section 5, we present the SWI-Ratio as an objective measure of task regularity, and further examine the connection between regularity and performance. In section 6, we transition to a general discussion of what this class of rule-based IE methods is able to learn from document collections. In section 7, we point out a number of limitations of current methods, focusing on what information is considered and how it is represented, how results are scored and presented, and the efficiency of training and testing. Finally, in section 8, we suggest improvements to address these limitations, and we provide experimental results that show that including grammatical information in the extraction process can increase regularity and performance.

2 Overview of algorithms and terminology

In this section, we present a brief review of the BWI approach to information extraction, including the formal problem statement, the algorithms used, and important terminology. We also present a simplified variant of the BWI algorithm, called SWI, which will be used to analyze BWI and related algorithms.

2.1 IE as a classification task

Most of the material in this section can be found in (Freitag, et. al, 2000). We present an abridged version here for convenience, starting with a review of relevant vocabulary. Each document can be broken up into a sequence of tokens. A token is one of three things: an unbroken string of alphanumeric characters, a punctuation character, or a carriage return. The problem of information extraction is to extract some number of tokens from a test document. To do this, we reformulate the IE problem as a classification problem. Instead of thinking of the problem as a string of tokens, we look at the problem as a function over boundaries. A boundary is the space between two tokens. Notice that a boundary is not something that is actually in the text (such as white space), but just comes about from the parsing of the text into tokens. We want to approximate two functions from a boundary to the binary set {0,1}: one function that is 1 iff the boundary is the beginning of a field to be extracted, and one function that is 1 iff the boundary is the end of a field. This transformation from a segmentation problem to a boundary classification problem is common and is also used by (Shinnou, 2001) to find word boundaries in Japanese text (since written Japanese does not use spaces).

These approximation functions are represented as sets of boundary detectors (or just detectors). A detector is a pair of token sequences, $\langle p, s\rangle$. A detector matches a boundary iff the prefix string of tokens, p, matches the tokens before the boundary and the suffix string of tokens, s, matches the tokens after the boundary. For example, the detector ⟨Who:, Dr.⟩ would match "Who: Dr. John Smith" between the ':' and the 'Dr'. Once the beginning and ending functions are approximated, extraction is performed by identifying the beginning and end of a field and extracting the text between the two points.

BWI separately learns two sets of boundary detectors for recognizing the beginnings and endings of a target field to extract. At each iteration, BWI learns a new detector, and then uses AdaBoost to reweight all the examples in order to focus on portions of the training set that the current rules are unable to match. Once this process has been repeated for a fixed number of iterations, BWI returns the two sets of learned detectors (*fore* detectors, F, and *aft* detectors, A), as well as a histogram over the lengths (number of tokens) of the target field, H.

2.2 Sequential Covering Wrapper Induction (SWI)

SWI uses the same basic framework as BWI with some slight modifications to the selection of rules. The basic algorithm for SWI can be seen in Figure 1. SWI generates F and A independently by calling the **`GenerateDetectors`** procedure. The **`GenerateDetectors`** procedure is a basic sequential covering rule learner. At each iteration through the while loop, a new boundary detector is learned, consisting of a prefix part and a suffix part. Like BWI, the best detector is generated by starting with empty prefix and suffix sequences. Then, the best token is found to add to either the prefix or suffix by exhaustively searching all possible extensions of length L. This is repeated, adding one token at a time, until the detector being generated no longer improves (as determined by the scoring function).

```
SWI: training sets S and E -> two lists of detectors and length histogram
  F = GenerateDetectors(S)
  A = GenerateDetectors(E)
  H = field length histogram from S and E
  return ‹F,A,H›

GenerateDetectors: training set Y(S or E) -> list of detectors
   prefix pattern p = []
   suffix pattern s = []
   list of detectors d

   while(positive examples are still uncovered)
     áp, sñ ← FindBestDetector()
     add áp, sñ to d
     remove positive examples covered by áp, sñ from Y
     p, s = []

   return d
```

Once the best boundary detector has been found, it is added to the set of detectors to be returned (either F or A). Before continuing, all the positive examples that were covered by the new rule are removed, leaving only uncovered examples. When this set of rules covers all the positive training examples, the loop ends, and the set of detectors that now cover all the positive training examples is returned.

A few things should be clarified at this point. There is a distinction between the actual text, which is simply a set of tokens, and the two training functions, which indicate whether a boundary is the start of end of a field to be extracted. When the algorithm removes the positive examples that are covered by a rule, the values in training function are set so that they are no longer positive or negative, but the actual sequence of tokens is not changed, as this would alter the data inappropriately.

There are a couple of key differences between BWI and SWI. First, as mentioned above, with sequential covering the training examples are altered by changing the set of positive examples. In BWI, the examples are reweighted but are never removed. Second, the scoring functions for the extensions and detector are slightly different. We propose two different versions of SWI. The basic SWI algorithm uses a simple greedy model. An extension or detector is scored simply by the number of positive examples that it covers without covering any negative examples. So, if a detector misclassifies even one example (i.e. covers a negative example), it is given a score of zero. We call this version Greedy-SWI. A second version of SWI, called Root-SWI, is also implemented and uses the same scoring function as BWI. Given the sum of the weights of the positive examples covered by the extension or detector, W^+, and the sum of the weights for the negative examples covered, W^-, the score is calculated to minimize training error (Cohen, et. al., 1999):

$$score = \sqrt{W^+} - \sqrt{W^-}$$

Notice that for Root-SWI these sums will just reflect the number of examples covered because the examples are all given the same weight. The final difference between BWI and SWI is that BWI terminates after a predetermined number of boosting iterations. In contrast, SWI terminates as soon as all of the positive examples have been covered. This turns out to be a key difference between the two different methods.

3 Experiments

In this section, we describe the data sets used for our set of experiments, our setup and methods, and the objective results of our tests. We provide an analysis of the results in the next section.

3.1 Data sets and IE tasks

In order to determine the differences between BWI, Greedy-SWI and Root-SWI we examine three algorithms on 15 different information extraction tasks from 8 different document collections. Many of these tasks are fairly standard and have been used in testing a variety of IE techniques. The tasks can be broken into three sets: natural (unstructured), partially structured, and highly structured. The breadth of these collections allows for a better analysis of the algorithms on a wider spectrum of tasks than has previously been studied. The natural text documents come from biomedical journal abstracts and contain little obvious structural information. The partially structured documents closely resemble natural text, but contain more high level structure and canonical formatting. These documents still consist primarily of natural language, but contain some regularities in formatting and annotations. For example, it is common for key fields to be preceded by an identifying label (e.g. "Speaker: Dr. X"), though this is not always the case. The highly structured documents consist of web pages that have often been automatically generated, and contain reliable regularities in formatting, including location of content, punctuation and style, and non-natural language such as HTML tags. The partially structured and highly structured document collections were obtained primarily from (RISE, 1998) and the natural text documents were obtained from the National Library of Medicine's MEDLINE abstract database (National Library of Medicine, 2001).

There are two collections of natural text documents, both made up of individual sentences from MEDLINE abstracts. The first collection consists of 555 sentences tagged with protein names, and their subcellular locations. These sentences were automatically selected and tagged from a larger collection of documents by searching for known proteins and locations contained in the Yeast-Protein-Database (*YPD*). Because of this labeling procedure, the labels are incomplete and sometimes inaccurate. Ray and Craven estimated a 10-15% error rate in labeling (Ray, et. al., 2001). The second collection consists of 891 sentences, containing genes and their associated diseases, as identified using the Online-Mendelian-Inheritance-In-Man (*OMIM*) database, and is subject to the same noisy labeling. Both these collections have been used in other research, including (Ray, et. al., 2001) and (Eliassi-Rad, et. al., 2001).

It should be noted that these document collections originally contained many sentences without labeled tuples because they were built for extracting relations. Therefore, only in cases where both elements of the relation were found was any label added. Because BWI is designed to extract only single fields, and not relations, we were forced to remove these documents, since many of them in fact contained examples of one of the desired fields, but as they were unlabeled, they presented BWI with contradictory information. For example, a good protein extraction rule would pull out many of the labeled proteins, but also many of the unlabeled proteins, and thus it would be considered a poor rule (and the unlabeled documents outnumbered the labeled documents by almost an order of magnitude). This still left a challenging set of information extraction tasks, but as a result our performance on these collections is not directly comparable to performance reported on the entire set of documents, when used to extract relations.

For the partially structured extraction tasks, we examine three different document collections. The first two were taken from RISE, and consist of 486 speaker announcements (*SA*) and 298 Usenet job announcements (*Jobs*). In the speaker announcement collection, four different fields were extracted: the speaker's name (*SA-speaker*), the location of the seminar (*SA-location*), the starting time of the seminar (*SA-*

stime) and the ending time of the seminar (*SA-etime*). In the job announcement collection, three fields were extracted: the message identifier code (*Jobs-id*), the name of the company (*Jobs-company*) and the title of the available position (*Jobs-title*).

We constructed the third partially structured documents from a collection of 630 MEDLINE journal citations. These documents contain full MEDLINE article citations for hip arthroplasty surgery, which consist of author/journal/publication info, MeSH headings (a standard controlled vocabulary of medical subject keywords), and other related information, including the text of the paper's abstract about 75% of the time. The task is to identify the beginning and end of the abstract text, if the citation includes one (*AbstractText*). The abstracts are generally large bodies of text, but without any consistent marker for the beginning or end, and the information that precedes or succeeds the abstract text also varies from citation to citation.

Finally, the highly structured tasks are taken from three different document collections, also in RISE: 20 web pages containing restaurant descriptions from the Los Angeles Times (*LATimes*), where the task is to extract the list of accepted credit cards (*LATimes-cc*); 91 web pages containing restaurant reviews from Zagat's Guide to Los Angeles restaurants (*Zagat's*), where the task is to extract the restaurant's address (*Zagats-addr*); and 10 web pages containing responses from an automated stock quote service (*QS*), where the task is to extract the date of the response (*QS-date*). Note that although these collections tend to contain a small number of documents, many of the individual documents are actually comprised of several separate entries (e.g. there are often multiple stock quote responses or restaurant reviews per web page).

3.2 Experimental setup

The performance of the different rule-based IE methods on these data sets was evaluated by three different standard metrics: precision, recall, and F1 (the harmonic mean between precision and recall). Each result presented is the average of 10 random 75/25 train/test splits. For all algorithms, a lookahead value, L, of 3 tokens was used. For the natural and partially structured texts (*YPD, OMIM, SA, Jobs,* and *AbstractText*), BWI's default wildcard set was used, and for the web pages (*LATimes, Zagat's,* and *QS*), the default lexical wildcards were also used. A graphical summary of our results can be seen in Figure 2. For complete numerical results, see Table 1 at the end of this paper.

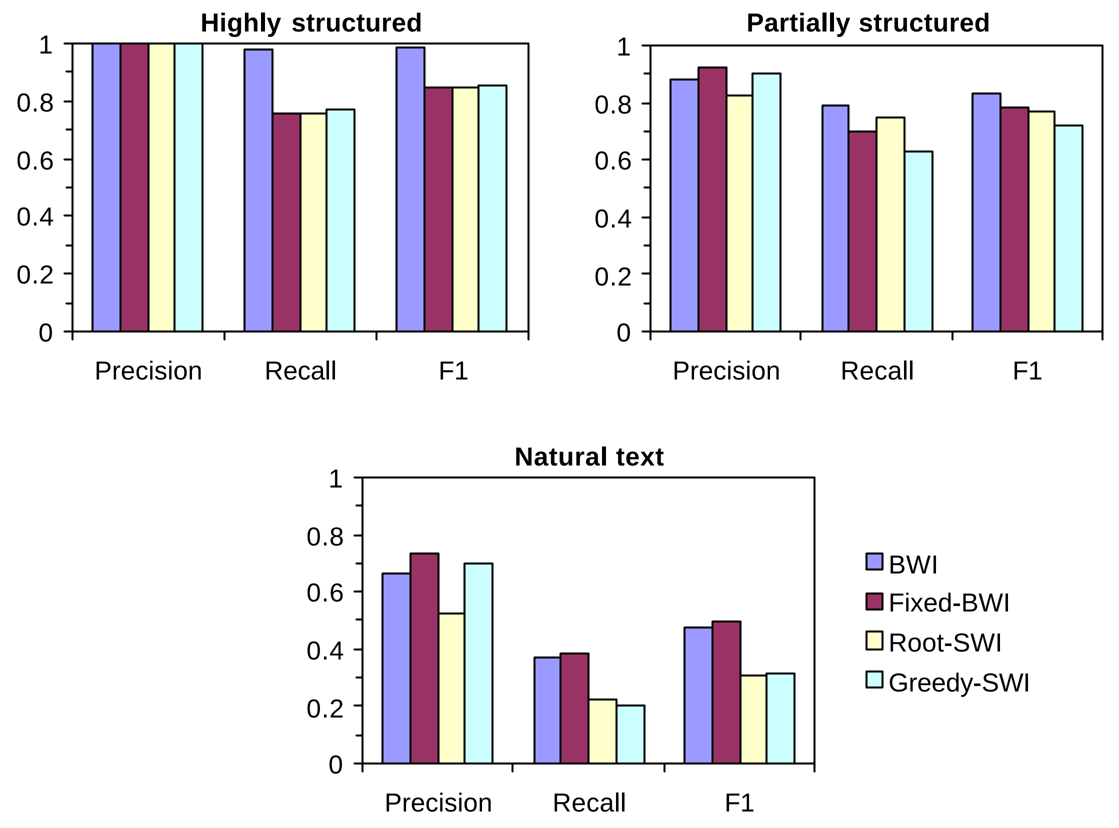

Figure 2: Performance of the four IE methods examined in this paper, separated by overall regularity of extraction task (indicated above each chart), averaged in each case over the set of tasks in the given domain.

To understand the differences between BWI and SWI, we performed four sets of tests. First, we ran BWI with the same number of rounds of boosting as was used in (Freitag & Kushmerick, 2000): for the, *YPD*, *OMIM*, *SA*, *Jobs*, and *AbstractText* document collections, 500 rounds of boosting were used; for the *LATimes*, *Zagat's*, and *QS* document collections, 50 rounds were used. Next, we ran Greedy-SWI and Root-SWI on the same tasks until all of the examples were covered. The actual number of iterations for which these algorithms ran varied across the different IE tasks. These numbers of iterations are presented in Table 2 at the end of the paper. Finally, we ran BWI for the same number of iterations that it took for Root-SWI to complete each task (e.g. 323 rounds for *YPD-protein*, 1 round for *QS-date*, etc.).

3.3 Experimental results

To investigate whether BWI outperforms the greedy approach of SWI, we compared Greedy-SWI and BWI. BWI appears to perform better than SWI—the F1 values for

BWI are higher for all the highly structured and natural text tasks studied. Greedy-SWI has slightly higher F1 performance on two of the medium structured tasks, but considerably lower performance on the other three. Generally, Greedy-SWI has slightly higher precision, while BWI tends to have considerably higher recall.

Given these results, the logical next step is determining what part of BWI's success comes from the difference in scoring functions used by the two algorithms, and what part comes from how many rules are learned. To explore the first of these differences, we compared Greedy-SWI and Root-SWI, which differ only by their scoring function. Greedy-SWI tends to have higher precision while Root-SWI tends to have higher recall. However, they have similar F1 performance, a result that is illustrated further by comparing BWI to Root-SWI. BWI still has higher F1 performance than Root-SWI in all but three medium structure tasks, despite the fact that they use an identical scoring function.

There are only two differences between BWI and Root-SWI. First, Root-SWI removes all positive examples covered after each iteration, whereas BWI merely reweights them. Second, Root-SWI terminates as soon as all positive examples have been covered, whereas BWI continues to boost for a fixed number of iterations. Examining Table 2 reveals that Root-SWI always terminates after many fewer iterations than were designated for BWI. In (Freitag, et. al., 2000), they examined BWI's performance as the number of rounds boosting increased, revealing that in many cases, results still improved even after all 500 iterations.

To investigate this idea further, we ran BWI for the same number of iterations as it took Root-SWI to complete each task (we call this method Fixed-BWI). Recall that the number of rounds Fixed-BWI runs for depends on the extraction task. Here the results appear to vary qualitatively with the level of structure in the domain. In the highly structured tasks, Fixed-BWI and Root-SWI perform nearly identically. In the medium structured tasks, Fixed-BWI tends to have exhibit higher precision but lower recall than Root-SWI, resulting in similar F1 values. In the natural text tasks, Fixed-BWI has considerably higher precision and recall than Root-SWI.

4 Analysis of experimental results

In this section, we discuss our experimental results from two different axes. We examine the effect of the algorithmic differences of the IE methods studied, and we look at how overall performance is affected by the inherent difficulty of the IE task.

4.1 Why BWI outperforms SWI

The set of experiments performed yield a detailed understanding of how BWI's algorithmic differences allow it to consistently achieve higher F1 values than SWI. The root of these differences is that BWI uses boosting to learn rules as opposed to set covering. Boosting has two complementary effects. First, boosting continually reweights all positive and negative examples to focus on the increasingly specific problems that the existing set of rules is unable to solve. This tends to yield high precision rules, as is clear from observing that Fixed-BWI consistently has higher precision than Root-SWI, even though they use the same scoring function and learn the same number of rules. While Greedy-SWI also has high precision, this is achieved by using a scoring function that doesn't permit any negative examples to be covered by any rules. This results in a lower recall than the other three methods, because it is hard to learn general rules with wide coverage without covering some negative examples.

Second, boosting allows BWI to continue learning even when the existing set of rules already covers all the positive training examples. Examples are merely reweighted after each iteration, rather than being removed entirely. In contrast, SWI is inherently limited in the total number of boundary detectors it can learn from a given training set. This is because every time a new detector is learned, all matching examples are removed, and training stops when there are no more positive examples left to cover. Since each new detector must match at least one positive example, the number of boundary detectors SWI can learn is at most equal to the number of positive examples in the training set (and usually many fewer detectors are learned because multiple examples are covered by single rules). The ability to continue learning rules means that BWI can not only learn to cover all the positive examples in the training set, but it can *widen the margin* between positive and negative examples, learning redundant and overlapping rules, which together better separate the positive and negative examples.

4.2 Rule-based IE methods and task difficulty

These two assets of boosting (learning specific rules, but learning more of them) together give BWI the consistent advantage in F1 performance over SWI that was observed empirically above. However, the difficulty of the extraction task clearly also has a pronounced effect on the performance of all four methods. We now turn from this general analysis to a specific investigation of each domain.

4.2.1 Highly structured

This set of tasks is by far the easiest for all methods to deal with—it's no coincidence that they're often referred to as "wrapper tasks" as learning the regularities in automatically generated web pages was precisely the problem for which wrapper induction was originally proposed as a solution. All methods have near-perfect precision, and SWI tends to cover all examples with only a few iterations.

Surprisingly, the same level of performance is not present for SWI's recall. While these tasks are all highly regular, the regularities learned on the first couple of iterations aren't the only important ones. Neither changing the scoring function from greedy to root nor changing the iteration method from set covering to boosting fixes this problem, as Root-SWI and Fixed-BWI have virtually identical performance to that of Greedy-SWI. However, because BWI continues to learn new and useful rules even after all examples are covered, it is able to learn secondary regularities that increase its recall and capture the differences that are missed by a smaller set of rules.

4.2.2 Medium structured

In this set of collections, we see the largest variation in performance by manipulating the components of the basic algorithms investigated. Changing Greedy-SWI's scoring rule to use BWI's root scoring function results in a tradeoff between increased recall and decreased precision and also a slightly higher F1. By allowing rules to cover some negative examples, more general rules can be learned that have broader coverage (higher recall), but also result in some negative examples being covered as well (lower precision). Root-SWI consistently terminates after fewer iterations than Greedy-SWI, confirming that it is indeed covering more examples with fewer rules.

Changing from set covering to boosting as a means of learning multiple rules also results in a precision/recall tradeoff with little change in F1. As mentioned earlier, boosting reweights the examples to focus in on the hard problems. This results in rules that are very specific (higher precision), but also very local (lower recall). Boosting also results in a slower learning curve (measured in performance vs. number of rules learned), because positive examples are often covered multiple times before all examples have been covered once. SWI explicitly removes all covered examples, focusing at each iteration on a completely new set of examples. The result is that Fixed-BWI can't learn enough rules to overcome its bias towards precision in the number of iterations Root-SWI takes to cover all the positive examples. However, if the full number of iterations is used, BWI compensates for this slow start by learning enough rules to ensure high recall without sacrificing this high precision. This ability to keep learning rules is only possible when using boosting, because set covering will always stop as soon as it runs out of examples to cover.

4.2.3 Natural text

Extensive testing of BWI on natural text has not been previously done, though it was the clear vision of Freitag and Kushmerick to pursue research in this direction. As can be seen in Figure 2, there are a number of qualitative differences in performance on the natural text documents when compared to the more regular collections previously studied. All the methods have considerably lower precision, despite the inherent high precision bias of the boundary detectors. All the methods also experience a large decrease in recall. This is mainly due to the fact that the algorithms often learn little more than the specific examples seen during training, which usually don't appear again in the test set. Looking at the specific boundary detectors learned, most simply memorize individual labeled instances of the target field, ignoring context altogether (i.e. the fore detectors have no prefix, and the target field as the suffix, and vice versa for the aft detectors). Changing SWI's scoring function from greedy to root results in a marked decrease in precision with only a tiny increase in recall. Negative examples are allowed to be covered, but the rules learned aren't sufficiently more general to increase coverage noticeably.

Unlike with the partially and highly structured tasks, Fixed-BWI has both higher precision and higher recall than Root-SWI. The high precision is explainable as before because reweighting focuses on difficult problems and learns specific solutions. In the more regular document collections, this also meant that the rules were too local. In this case, however, it actually increases recall. The reason is that while already covered examples are down-weighted during boosting, they are not removed entirely as with SWI. Thus, BWI remains sensitive to the accuracy and coverage of all the rules it learns. In contrast, SWI quickly eliminates the regularities that can be easily found, and then begins covering the examples one at a time. Thus, it completely ignores the performance of new rules on already covered examples. This problem is compounded by the fact that BWI reweights both positive and negative examples, while SWI never removes negative examples. This is particularly troublesome if we consider that these data sets were labeled automatically, and are partially inaccurate (as mentioned in 3.1). SWI only covers positive examples once, and it is more sensitive to negative examples, so the perceived difference in regularity between the training and test set becomes more pronounced as the algorithm continues to eliminate positive examples.

Unlike in the more structured document collections, allowing BWI to run for the full 500 iterations caused it to perform worse than Fixed-BWI, suggesting that the extra iterations result in overfitting. Because of the irregularity of the data, Fixed-BWI runs for a larger number of iterations than on easier tasks, and thus the extra benefit of continuing to boost is diminished. Furthermore, these last rounds of boosting tend to concentrate on only a few highly weighted examples, meaning that unreliable rules are learned, which are more likely to be artifacts of the training data than true regularities.

5 Quantifying task regularity

In addition to the observed variations resulting from changes to the scoring function and iteration method of BWI and SWI, it is clear that the inherent difficulty of the extraction task is also a strong predictor of performance. Highly structured documents such as those generated from a database and a template are easily handled by all the algorithms we tested, whereas natural text documents are uniformly more difficult. It is thus interesting to investigate methods for quantifying the regularity of a document set beyond the broad classes used thus far.

5.1 SWI-Ratio as a measure of task regularity

We propose that a good measure of the regularity of an extraction task is the ratio of the number of iterations SWI takes to cover all the positive examples to the total number of positive examples in the task. We call this measure the *SWI-Ratio*, for obvious reasons. In the most regular limit, a single rule would cover an infinite number of documents, and thus the SWI-Ratio would be 1/ = 0. At the other extreme, in a truly irregular set of documents, SWI would have to learn a separate rule for each positive example (i.e. there would be no generalization possible), so for N positive examples, SWI would need N rules, for an SWI-Ratio of N/N = 1. Notice that since each SWI rule must cover at least one positive example, the number of SWI rules learned for a given document set will always be less than or equal to the total number of positive examples, and thus the SWI-Ratio will always be between 0 and 1, with a lower value indicating a more regular extraction task.

SWI-Ratio is a meaningful and well-motivated measure of task regularity for several reasons. First, it is a relative measure with respect to the amount of text, so one can use it to compare regularity of small and large documents alike. Second, it is general and objective, because SWI is such a simple algorithm (essentially equivalent to greedy set covering), and since SWI's scoring rule doesn't allow any negative examples to be covered, it is an unbiased account of how many non-overlapping rules one would need to actually cover every positive example without covering any negative examples. Finally, it is easy and uncontroversial to attain. There are no free parameters to set (other than the lookahead for extending boundary-detector rules, which we fixed across all our tests), and it is simple and relatively efficient to run SWI before or while running any other technique.

5.2 Performance and task regularity

In addition to the clear differences in performance between the three classes of extraction tasks presented in the previous sections, there is also wide variation within each document collection. Although the results in Figure 2 are easily interpreted, we must be careful not to forget that these values represent averages over several extraction tasks. However, using the SWI-Ratio, we can now compare algorithms on a per-task basis.

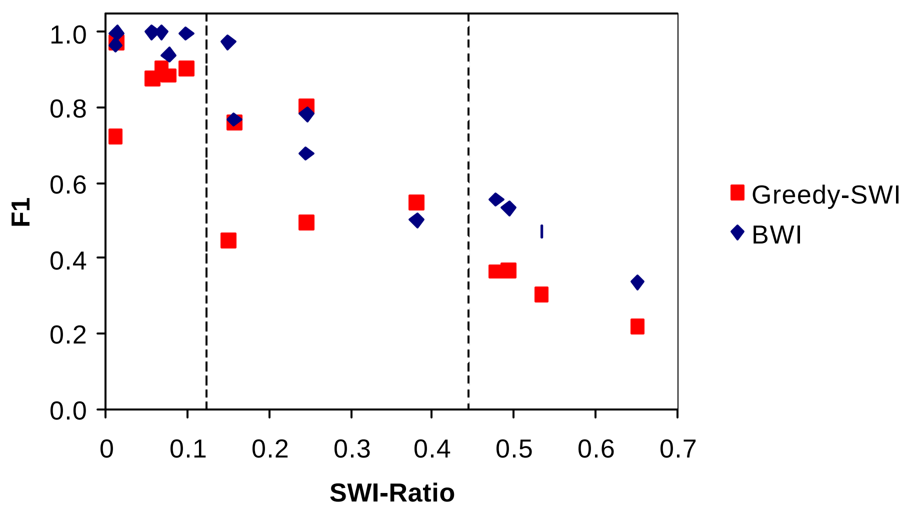

Figure 3: F1 performance for BWI and Greedy-SWI on the 15 different extraction tasks, plotted vs. the SWI-Ratio of the task, with separations between highly structured, partially structured, and natural text.

This will reveal in greater detail both how performance is related to task difficulty, and whether BWI reliably outperforms SWI.

In Figure 3, the F1 of Greedy-SWI and BWI are plotted for each task versus its SWI-Ratio. As noted in section 4, BWI performs better than Greedy-SWI on all but two tasks and the difference between the F1 for those two tasks is small. By plotting the two algorithms' performance versus the SWI-Ratio, we can examine how the two algorithms perform as the regularity of the documents decreases. There is a consistent decline in the performance as the regularity of the document decreases for both algorithms. Also, there does not seem to be a significant divergence in performance between the two algorithms as the regularity of the task decreases. So, although BWI appears to reliably outperform SWI, both these algorithms are still limited by the regularity of the text.

6 What rule-based IE methods are learning

Having investigated BWI as an important new IE technique, and having understood its specific advantages over comparable strategies, we now turn to the more general issue of rule-based IE methods and their performance as a class. Specifically, we investigate the relevant information that these methods do and do not appear to be learning during training.

As has been seen in the experimental section and in numerous other papers, rule-based systems perform quite well with fairly structured tasks. If the target field to be extracted is canonically preceded or followed by a given set of tokens (or tokens of a distinct type, matched by the wildcards used), this context is readily learned by boundary detectors. This is a common occurrence in highly structured and partially structured texts, where fields are often preceded by identifying labels (e.g. "Speaker: Dr. X"), or followed by the same pieces of information (e.g. in the Zagat's surveys, the restaurant address is almost always followed by its telephone number, which is easily distinguished from the rest of the text).

Surprisingly, there is a good deal of this type of regularity to be exploited even in more natural texts. For example, when extracting the locations where proteins are found, "located in" and "localizes to" were found to be common prefixes, and were learned early on by BWI. This is reflects the fact that people often refer to a given type of information in a specific context, and thus specific lead-in words or following words are common and used in many instances.

While rule-based IE methods are primarily designed to identify the contexts in which target fields occur, it is important to realize that BWI and other methods are also learning some things about the regularities that occur in the fields being extracted. In the case of BWI, boundary detectors extend into the edge of the target field as well as into the local context, so the fore detectors often learn what the first few tokens of the target field look like (that is, if the field tends to have a regular beginning), and the aft detectors often learn what the last few tokens look like.

With short fields, this means that in many cases, individual boundary detectors are doing nothing more than memorizing instances of the target field. This is often the case in the MEDLINE articles, where specific gene names, protein names, etc. get memorized when their context isn't otherwise helpful. However, with longer fields, there is still important information learned. For example, in the MEDLINE citations, the abstract text often starts with phrases like "OBJECTIVE: ", or "We investigated…"

In addition to learning the canonical starting and ending tokens in the target field, BWI learns a length distribution of the number of tokens in the extracted field. Specifically, field lengths are recorded for each training example, and then this histogram is normalized into a probability distribution once training is complete. In BWI, this is the only piece of information that explicitly ties the fore detectors and aft detectors together. Nevertheless, this information can be extremely useful. When BWI tests on a new document, it first notes every fore and aft detector that fire, and then must pair them up to find specific token sequences. BWI only keeps a match that is consistent with the length distribution. Specifically, it will drop matches whose fore and aft boundaries are farther apart or closer together than has been seen before during training, and given two overlapping matches of equal detector confidence, it will prefer the match whose length has been seen more times.

In all but the most regular IE tasks, a single extraction rule is insufficient to cover all the positive examples, so it is necessary to learn a set of rules that capture different regularities. An important piece of information to capture, then, is the relative prominence of these different types of patterns, because it distinguishes the general rules from the exceptional cases. In the case of BWI, this information is learned explicitly by assigning each boundary detector a confidence value, which is proportional to the total weight of positive examples covered by the rule during training, and is thus broadly reflective of the generality of the rule. In SWI, rules that cover the most examples are learned first, so there is a ranking present as well.

7 Limitations of BWI (what rule-based IE methods *aren't* learning)

While BWI and similar methods currently achieve high levels of performance on many document collections, there is substantial room for improvement. There are useful pieces of information that most IE methods currently ignore that might increase the apparent regularity of the extraction task were they available. There are also important facts and clues about many extraction tasks that the current representations of such methods are unable to capture. Finally, there are issues with the speed and efficiency of current methods, as well as how matches are scored and presented, that limit the usefulness of these systems. We investigate each of these areas in more detail in this section.

7.1 Representation

BWI learns sets of fore and aft boundary detectors and a histogram of the lengths of the fields. These boundary detectors are short sequences of specific words or wildcards that directly preceed or succeed the target field or that are found within the target field. As a result of this representation, there are valuable sources of regularity that cannot be captured with the current representation, such as a more detailed model of the field to be extracted, the global location of the field within the document, and any structural information such as that exposed by a grammatical parse or an HTML/XML document tree.

7.1.1 Weak model of the content being extracted

Clearly an important clue in finding and extracting a particular fact is recognizing what relevant fragments of text look like. As mentioned in the previous sections, BWI learns to recognize the canonical beginnings and endings of the target field, as well as a distribution over the length of the extracted field. However, both these pieces of information are learned in a superficial manner, and much of the regularity of the field is ignored entirely.

Currently, BWI normalizes its frequency counts of field lengths in the training data without any type of smoothing or binning. This means that any candidate field to be extracted whose length is not precisely equal to that of at least one target field in the training data will be categorically dismissed, no matter how compelling the boundary detection. While this does not tend to cause problems for very short fields (because there are only a few possible lengths), it is a tremendous problem for fields with a wide variance of lengths.

For example, in the MEDLINE citations task, the abstract texts to be extracted have a mean length of 230 tokens, with a standard deviation of 264 tokens. With only 462 positive examples of abstracts (in 630 citations), it’s not surprising that many reasonable token lengths are simply never seen. When running our experiments on this document collection, we had to set BWI to ignore all length information. Ignoring the length information resulted in F1 increasing from .1 to .97. This problem could obviously be remedied by smoothing the distribution somehow, or by at least binning the lengths prior to normalizing.

Besides the length distribution, the only other information BWI learns about the content of the target field comes from boundary detectors that overlap into the field. While for short fields this can mean memorizing entire instances of the target field, in most cases there is more information about the canonical form of the target field that could be captured and exploited. There has been a great deal of work on modeling fragments of text to identify them in larger documents, most under the title of *Named-Entity Extraction*. For example, (Baluja, 1999) report impressive results on finding proper names in free text, using only models of the names themselves, based on capitalization, common prefixes and suffixes, and other word-level features. NYMBLE is another notable work in this field (Bikel, 1997). Perhaps combining these field-finding methods with BWI's context-finding methods would yield superior performance in both domains.

7.1.2 Limited expressiveness of boundary detectors

Boundary detectors were designed to capture the flat, local context of the field to be extracted by learning short sequences of surrounding tokens. This is the source of their success and their failure. On the one hand, they are good at capturing regular, sequential information around the field to be extracted, resulting in high precision. On the other hand, the high precision design of current boundary detectors tends to result in rules having poor recall (Freitag et. al., 2000). This is not so much a problem for highly structured tasks, because there are only a few regularities that need to be captured. However, these rules are less effective in partially structured and natural texts, because regularities in context are less consistent and reliable. This results in many rules being learned that only cover one or a few examples.

Another limitation of existing boundary detectors is that they completely ignore any grammatical structure of sentences and a great deal of information present in HTML/XML documents. That is, boundary detectors can only capture information about the tokens that appear directly before or after the target field in the linear sequence of tokens in a document. BWI and similar methods do not contain any information about the parent nodes, siblings, or child position in the grammar or XML/HTML tree in which they implicitly belong. In addition, the boundary detector format does not allow BWI to take advantage of part-of-speech information, and other similar features that have been shown to contain important sources of regularity.

While context information is often the most relevant for extracting fields, global information about the relative position of the field within the context of the entire document can also be important. For example, despite BWI's impressive performance on the abstract text data set, the most obvious clue to finding an abstract in a MEDLINE citation is, "look for a big block of text in the middle of the citation." There is no way to capture this global location knowledge using BWI's existing representation. Additional knowledge that BWI is unable to learn or use includes, "the field is in the second sentence of the second paragraph if it's anywhere," "the field is never more than one sentence long," "the field doesn't have any line breaks inside it," and so on.

7.2 Scoring

Every match that BWI returns when being tested has an associated confidence score, which is the product of the confidences of the fore and aft boundary detectors along with the probability of the resulting field length. The scoring of the boundary detectors is important in identifying the most useful rules and also trying to assess how well the rules will perform. This score may be used for pruning less productive rules and/or thresholding to get a precision/recall tradeoff.

While the scores are a good measure of the likelihood with which a proposed match is in fact correct, these scores are not as meaningful or useful as actual match probabilities would be. First of all, the confidence scores are unbounded positive real numbers, and thus it is difficult to assess the relative confidence of matches in different document collections which have a large dynamic range. This also makes it hard to set absolute confidence thresholds for accepting matches.

Second, it's hard to compare the confidence of full matches and partial matches, because the components of the confidence score have dramatically different ranges. This obscures the "decision boundary" of the learned system, which would otherwise be an important target for improving performance. Finally, without probabilities, it is difficult to use the match scores in a larger framework, for example Bayesian evidence combination or decision-theoretic action planning. The basic problem is that while the scores are useful for stating the relative confidence of two matches in the same document collection, they are not useful for comparing matches across collections, nor are they useful for providing an absolute measure of confidence. Unfortunately there is much demand for the later two cases.

Another problem with using unbounded positive numbers to score boundary detectors is that no distinction can be made between a fragment of text that has appeared numerous times in the training set as a negative example, a field that has never been seen labeled one way or the other, and a field where one boundary detector has confidently fired. In other words, there is no way to tell a "confident rejection" from a "near miss." This "negative space" is as important as the positive space for communicating confidence in extracting fragments of text, but it has been completely collapsed into a single point (confidence = 0).

BWI also implicitly assumes that there is only one type of field being extracted from a given data set. Most documents contain multiple pieces of related information to be extracted, and the position of one field is often indicative of the position of another. However, BWI can only extract fields one at a time, and entirely ignores the relative position of different fields. For example, in the speaker announcements, even though there are four fields being extracted, they are all trained and tested on independently, and the presence of one field is in no way considered as usefully predictive of the location of another. This also makes it difficult to extend methods like BWI to extract relationships in text, which is often as important as extracting individual fields (Muslea, 1999). Furthermore, it is inconvenient for real-world settings, where ideally one should be able to label and train on a single document at a time, with multiple fields labeled in each document, rather than going through every document in the set multiple times.

7.3 Efficiency

In addition to the clear need to develop high performance IE systems, speed and efficiency of training and testing is also a critical component for making these systems practical for solving real-world problems.

7.3.1 BWI is slow to train and test

Even on modern workstations, training BWI on a single field with even a few hundred documents can take several hours, and testing can take several minutes. The slowness of training and testing also make using larger document collections or larger lookeahead for detectors prohibitive, even though both might be necessary to achieve high levels of performance on complex tasks. This also inhibits reliable comparison of different IE methods, and makes BWI unsuitable for the engine of an interactive system, such as with iterative human labeling during active learning. There are a number of factors that make BWI slower than may be necessary.

The innermost loop of training BWI is finding extensions to boundary detectors. This is done in a brute-force manner, up to a specified lookahead parameter, L, and repeated until no better rule can be found. Finding a boundary detector extension is thus exponential in L, because every combination of tokens and wildcards is enumerated and scored, which makes even modest lookahead values prohibitively expensive. As a result, a value of L=3 is normally used to achieve a balance between efficiency and performance. Freitag et. al. note that while this was usually sufficient, there were times when a value up to L=8 was required to achieve the best results. Training with such a high lookahead is usually prohibitively slow, however, and thus the "local context" learned tends to be very local indeed.

Although testing is considerably faster then training, it too is inefficient and slow. One technique for improving testing speed is to compress the set of boundary detectors learned during training, before applying them to a set of test documents. For example, one could eliminate redundancy by combining duplicate detectors, or eliminating detectors that are logically subsumed by a set of other detectors (since whenever one would match, so would the other). This is a practice used by (Cohen, 1999) in SLIPPER, as well as by (Ciravegna, 2001) in $(LP)^2$ for partially structured tasks, but it's doubtful how useful it would be in less regular document collections, which unfortunately are the ones in which a large number of detectors have to be learned. There are also more sophisticated ways to compress a set of rules, many of which are somewhat lossy (Margineantu, 1997; Yarowsky, 1994). However a tradeoff between speed and accuracy should always be evaluated.

7.3.2 BWI can't learn incrementally

Once BWI has been trained on a given set of labeled documents, if a few more documents are labeled and added, there is currently no way to avoid training on the entire set from scratch again. In other words, there is no way to learn some extra marginal information from a few additional documents, even though the vast majority of what will be learned in the second run will be identical to what was learned before. This is more a problem with boosting than with BWI specifically, because the reweighting that occurs at each round depends on the current system's performance on all the documents, so even adding a few extra documents might mean that the training takes a very different direction. Nevertheless, it is clearly desirable to be able to add documents to a data set as they become labeled, and not to have to completely retrain each time.

7.3.3 BWI doesn't know when to stop boosting

While BWI's ability to boost for a fixed number of rounds instead of terminating once all positive examples have been covered is clearly one of its advantages, it's difficult to know how many rounds to prescribe. Depending on the difficulty of the task, a given number may be insufficient to learn all the exceptions, or it may be overkill and lead to overfitting or massive redundancy. Ideally, BWI would be able to continue boosting for as long as useful, and then shut itself off when it deems that continuing to boost would do more harm than good. This could also serve as an alternative to having to go back and prune the trained rule list, because the rules learned in the final iterations are likely to be the least reliable and useful.

Cohen and Singer address the problem of when to stop boosting directly in their design of SLIPPER by using internal five-fold validation on a held out part of the training set (Cohen, 1999). For each fold, they first boost up to a specified maximum number of rounds, testing on the held out data after each round, and then they select the number of rounds that lead to the lowest average error on the held out set, finally training for that number of rounds on the full training set. This is certainly a reasonable thing to do, and it has the advantage of being sensitive to the difficulty of the task, but it also has several drawbacks. First of all, it makes the already slow training process six times slower, when presumably at least part of the motivation behind finding an optimal number of rounds is to avoid spending more time than necessary during training. Second, they replaced the free parameter of the actual number of boosting rounds with another free parameter for the maximum number of rounds to try during cross-validation. Setting the value too low will mean that the best number of rounds is never found, while setting the number of rounds too high will mean much wasted effort.

Perhaps it would be possible to use BWI's own detector confidences to tell when continuing to boost is futile or harmful. Since detector scores tend to decrease as the number of boosting rounds increases, it might be possible to set an absolute or relative confidence threshold below which to stop boosting, or to determine when the curve has flattened out and is only picking up exceptional cases one at a time. For example, (Ciravegna, 2001) prunes rules that cover less than a specified number of examples, because they are unreliable, and generally more likely to cause spurious matches than actual ones.

8 Extending rule-based IE methods

In this section, we present several suggestions for advancing the research agenda of rule-based IE methods as a whole. We concentrate on identifying new sources of information to consider, constructing new representations for handling them, and producing more meaningful output from the system. In some cases, we present preliminary results that appear to corroborate our hypotheses. In other cases, we cite existing work by other researchers that we believe represent steps in the right direction. Our hope is that this will provide a clearer organizational structure for existing research, as well as an inspiration for novel projects.

8.1 Exploiting grammatical structure of sentences

Section 7.1.2 discusses the importance of using grammatical structure in natural language or hierarchical structure in HTML and XML documents. Ray and Craven have taken an important first step in this direction (Ray, et. al., 2001) by preprocessing natural text with a shallow parser, and then flattening the parser output by delimiting sentences into typed phrase segments. The text is then marked up with this grammatical information, and is used as part of the information extraction process (they use Hidden Markov Models, but this technique is generally applicable). For example, using XML tags to represent these phrase segments, they construct sentences from MEDLINE articles such as:

<NP_SEG>**Uba2p**</NP_SEG> <VP_SEG>**is located largely**</VP_SEG>
<PP_SEG>**in**</PP_SEG> <NP_SEG>**the nucleus**</NP_SEG>**.**

While the parses produced aren't perfect, and the flattening often further distorts things, this procedure is fairly consistent in blocking out noun, verb, and prepositional phrases. An information extraction system can then learn boundary detectors that include

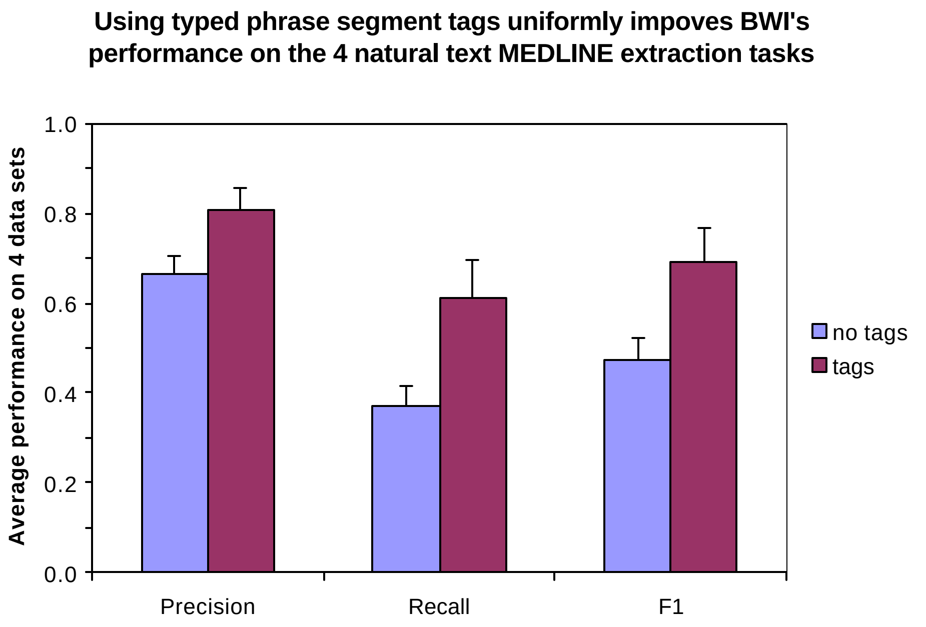

Figure 4: Performance of BWI averaged across the four natural text extraction tasks, with and without the use of typed phrase segments. Shown with standard error bars.

these tags, allowing it, for example, to represent the constraint that proteins tend to be found in noun phrases, and not in verb phrases. Ray and Craven report that their results "suggest that there is value in representing grammatical structure in the HMM architectures, but the *Phrase Model* [with typed phrase segments] is not definitively more accurate."

In addition to the results presented in the experimental section, which include Ray and Craven's data sets without any grammatical information, we also experimented on their data using this segmental information in order to see if there were valuable regularities expressed in this extra content that could be exploited by rule-based IE methods. We used BWI to extract all the individual fields in the two relations that Ray and Craven study (proteins and their localizations, genes and their associated diseases). We ran identical tests with and without the XML tags as shown above. We found that including the tags uniformly and considerably improved both precision and recall for all four extraction tasks (Figure 4). In fact, all four tasks saw double-digit percentage increases in precision, recall, and F1, with an average increase of 21%, 65%, and 46% respectively.

The fact that precision improves when using the phrase segment tags means that BWI is able to use this information to reject possible fields that it would otherwise return. The fact that recall also improves suggests that having segment tags helps BWI to find fields that would otherwise miss. The combination of these results is somewhat surprising. For example, while not being a noun phrase may be highly correlated with not being a protein, the inverse is not necessarily the case (i.e. there are plenty of non-protein noun phrases in MEDLINE articles). Thus we hypothesized that including typed phrase segment information was actually regularizing the extraction task, enabling rules to gain greater positive coverage without increasing their negative coverage.

Using the SWI-Ratio we proposed earlier in the paper, we were able to quantify and address this hypothesis. We ran SWI on all four extraction tasks with and without tags, and compared their resulting SWI-Ratios (Figure 5). As expected, when we included the tags, we saw double-digit percentage decreases in the SWI-Ratio of all four tasks, with an average reduction of 21%. Recall that a lower SWI-Ratio represents a more regular domain, because it means that the same number of positive examples can be perfectly covered with fewer rules. We conclude that including this type of grammatical information, even with existing rule-based IE methods, represents a considerable advantage for both accuracy and coverage, and is worth investigating in more detail.

The success of including this limited grammatical information immediately raises the question of what additional grammatical information could be used, and how helpful it might be. It's not hard to imagine that exploiting regularities in field context such as argument position in a verb phrase, subject/object distinction, and so on would be valuable. For example, Charniak has shown that probabilistically parsing sentences is greatly aided by conditioning on information about the linguistic head of the current phrase, even if it's several tokens away in the flat representation (Charniak, 2001). This suggests that such information is an important source of regularity, which is exactly what rule-based IE methods are designed to exploit.

Unfortunately, the existing formulation of boundary detectors is not well equipped to represent or handle such information. While merely inserting XML tags to represent typed phrase segments proved useful, such an approach is unprincipled, as it combines meta-level information with the text itself. Ideally, the representation of

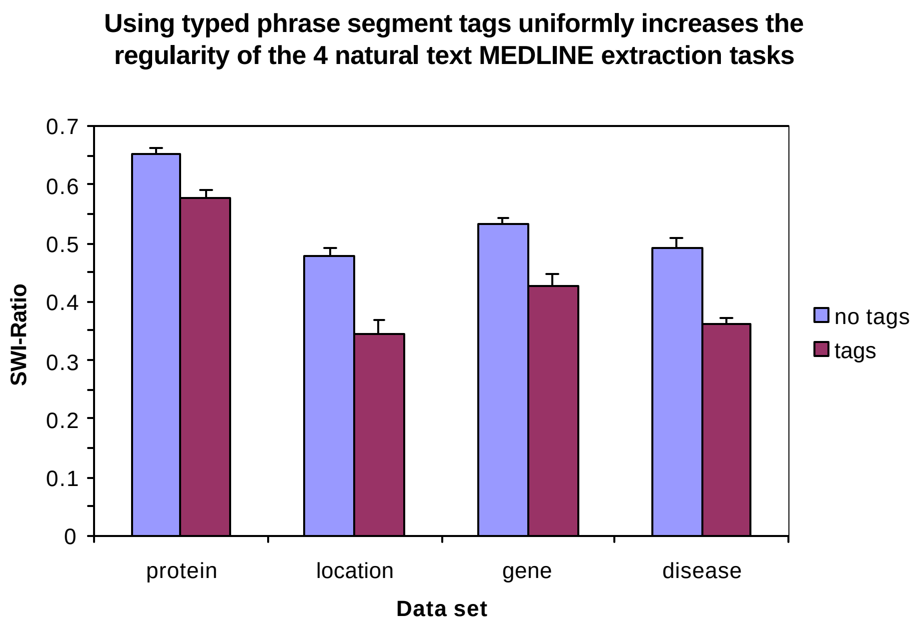

Figure 5: Average SWI-Ratio for the four natural extraction tasks, with and without the use of typed phrase segments. Shown with standard error bars.

boundary detectors should be extended to capture and exploit regularities in implicit higher-level information as well as explicit token information in a document. In the case of using typed phrase segments, we were able to increase performance while maintaining the simple, flat representation currently used. However, this approach can only be taken so far.

8.2 Handling XML / HTML structure and information

Just as the grammatical structure of a sentence presents opportunities for exploiting structural regularities, the hierarchical structure and attribute information defined by an XML or HTML document also contain important clues for locating target fields. However, this information is essentially lost when an XML document is parsed as a linear sequence of tokens instead of into a Document Object Model (DOM). For example, tags like `<tag>` or `</tag>` that should be treated as single tokens are instead broken into pieces. More problematic, however, is the fact that many tags contain namespace references, and attribute-value pairs inside the starting tag, such as `<name:tag att1="val1" att2="val2">`. When using flat tokenization, lookahead becomes a serious problem, and there is no way to intelligently generalize over such tags (e.g. match a tag with the same name, require specific attributes/values, etc.).

(Muslea, et. al., 1999) have made some important contributions to solving this problem with STALKER, which constructs an "embedded catalog" for web pages (a conceptual hierarchy of content in a document) that allows them to take some advantage of global landmarks for finding and extracting text. They accomplish this by use of the "Skip-To" operator, which matches a boundary by ignoring all text until a given sequence of tokens and/or wildcards. While the token sequences learned with Skip-To operators arte similar to the boundary detectors used in BWI, they can be combined sequentially to form an extraction rule that relies on several distinct text fragments, which can be separated by arbitrary amounts of intermediate text.

Rules in STALKER are learned by starting with the most simple Skip-To rule that matches a given positive example and greedily adding tokens and new Skip-To operators as necessary to eliminate covering any negative examples. This approach avoids the exponential problem of the exhaustive enumeration of boundary detectors used by BWI. The tradeoff, however, is that the resulting boundary detectors only work for pages with a highly regular and consistent structure, and the token sequences learned tend to be shorter and more specific. Furthermore, using Skip-To only approximates the information available in a true DOM representation.

Another approach to exploiting the structural information embedded in web pages and XML documents can be found in (Yih, 1997), which uses hierarchical document templates for IE in web pages, and finds fields by learning their position within the template's node tree. The results presented are impressive, suggesting that this is useful information to have, but currently the document templates must be constructed manually from web pages of interest, because the hierarchies in the templates are more subjective than just the HTML parse. This may be less of a problem with XML documents, but only if the tag-based structure corresponds in some relevant way to the content structure that is necessary to exploit for information extraction. Similar results are presented by Liu, et. al. in "XWRAP", a rule-based learner for web page information extraction that uses heuristics like font size, block positioning of HTML elements, and so on to construct a document hierarchy and extract specific nodes (Liu, 2000).

8.3 Extending the expressiveness of boundary detectors

One quick solution to the problem of incorporating more grammatical and structural information into the existing rule-based IE framework is the development of a more sophisticated set of wildcards. Currently, wildcards only cover word-level syntactic classes, such as "all caps", "begins with a lower-case letter", "contains only digits", and so on. While these are useful generalizations over matching individual tokens, they're also extremely broad. A potentially middle ground might be developing wildcards that match words of a given linguistic part of speech (e.g. "noun"), a given semantic class (e.g. "location/place"), or a given lexical feature (e.g. specific prefix/suffix, word frequency threshold, etc.).

In principle, such wildcards could be built and used with the existing BWI framework, and would be able to capture regularities that are currently being learned one instance at a time. However, with many of these new types of wildcards, constructing either a predefined set of allowable words or a simple online test for inclusion would not be feasible, and instead we would have to rely either on preprocessing using existing NLP systems (as Ray and Craven did), or on grammatical/lexical classifiers that could be used as-needed. Clearly efficiency would become an important issue, particularly during training when such generalizations would have to be repeatedly posed and checked.

Encouraging results in this direction are already available. For example, Ciravegna's $(LP)^2$ system uses word morphology and POS information to generalize the specific rules it initially learns (Ciravegna, 2001), a process essentially equivalent to using lexical and POS wildcards respectively. Ciravegna's results are comparable to other state of the art methods, including BWI. While $(LP)^2$ also uses rule correction and rule pruning, Ciravegna attributes "the use of NLP for generalization" as being the most responsible for the performance of the system. Another clever use of such generalizations can be found in RAPIER (Califf, et. al., 1999), which uses WordNet (Miller, 1995) to find hypernyms (a semantic class of words to which the target word belongs), and uses them in its learned extraction rules. (Yarowsky, 1994) also reports that including semantic word classes like "weekday" and "month" to cover sets of tokens improves performance for lexical disambiguation, suggesting that there are indeed regularities to be exploited at this level of generality.

HTML/XML wildcards that would allow for generalizations such as any "font-formatting tag" or any "table cell with a specified background color" could also be employed. These generalizations would not only help increase recall without sacrificing precision, they might help solve a common problem in wrapper induction, where small changes to a web page "break" the current learned detectors, and the system has to be retrained. For example, if a web page changes the font color of a key piece of information, but the boundary detector has learned a more general rule that identifies any text with a non-standard font color, it may well continue to fire correctly, whereas a more specific rule would no longer match at all.

8.4 Turning BWI match scores into probabilities

While improving the performance of IE methods is obviously an important goal, any progress made will be limited in its usefulness by the output of the system, and thus this is another critical target for further research. As mentioned in the previous section (7.2), BWI and similar methods do not yet attach probabilities to the matches they return. Probability is the *lingua franca* for combining information processing systems, because probabilities have both absolute and relative meaning, and because there are powerful mathematical frameworks for dealing with them, such as Bayesian evidence combination and decision theory.

Thus, it is worth asking the question, can BWI's confidence scores be transformed into probabilities? This question really comes in two pieces. First, are BWI's confidence scores meaningful and consistent? That is, does BWI produce incorrect matches with high confidence, or does its accuracy increase with its confidence? Second, can BWI learn to correlate its own confidences with the results returned, and thus automatically calibrate its scores into probabilities?

The first question (are BWI's scores consistent) can be answered empirically, by training and testing BWI on different document collections. Ideally, the majority of BWI's incorrect predictions would have low confidence compared to its correct predictions. Such a distribution would be desirable for two reasons. First, it would mimic a true probability distribution, where a higher score is correlated with a higher chance of being correct. Second, it would allow users to set a confidence threshold above which to accept the computer's guesses, and below which to flag for human follow up.

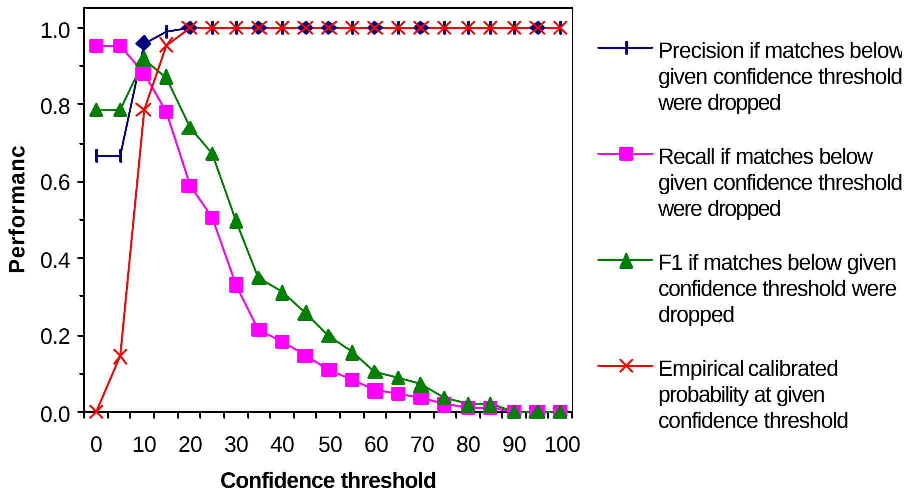

Figure 6: Precision, recall, and F1 of BWI on *AbstractText* task vs. confidence threshold below which to ignore matches. Shown with empirical probability of matches within each confidence interval.

Our results suggest that BWI's scores are fairly consistent and amenable to a threshold setting for regular tasks, but that on hard tasks it is difficult to separate BWI's correct and incorrect predictions based only on its ma tch confidences. For the *AbstractText* task, confidence scores ranged from 0 to 100, but all of BWI's incorrect guesses had confidence scores below 20 (Figure 6). This means that if BWI merely ignored any of its results below a confidence threshold of 20, it would obtain perfect precision. However, setting a confidence threshold of 20 would result in 59% recall, compared with 95% recall with a threshold at 0, because correct answers would also be pruned. In this case, a lower confidence threshold will increase F1, because most of the incorrect answers can be pruned without eliminating correct guesses, as can be seen in the F1 curve in Figure 6, which peaks at a confidence threshold of 10. This means the confidence scores are behaving roughly as probabilities should, because as the confidence increases, so does the fraction of correct guesses. Note that the precision for this task is lower than reported in Table 1 because we considered all of BWI's matches, whereas normally BWI eliminates overlapping matches, keeping the candidate with higher confidence.

Unfortunately, this desirable behavior does not appear to hold up for more difficult tasks. On the protein task, considered the most difficult both by performance

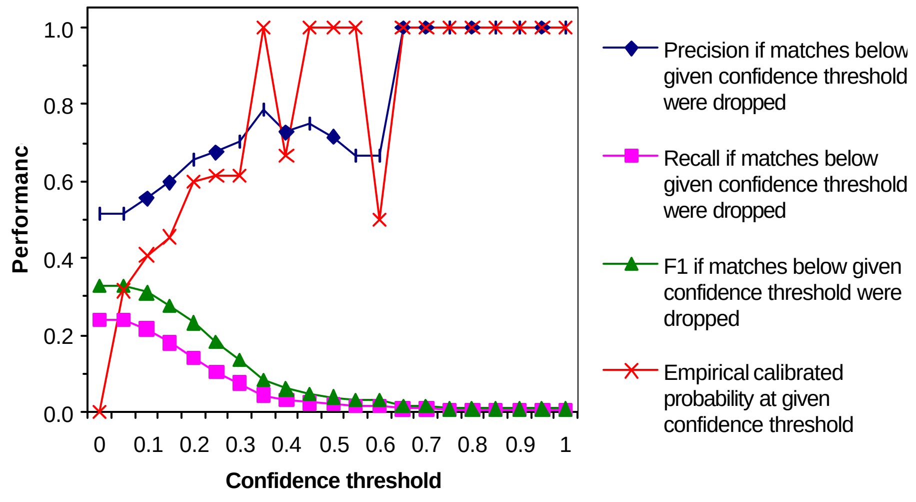

Figure 7: Precision, recall, and F1 of BWI on *YPD-protein* task vs. confidence threshold below which to ignore matches. Shown with empirical probability of matches within each confidence interval.

and by SWI-Ratio, there is no clean way to separate the correct and incorrect guesses (Figure 7). The highest confidence negative match gets confidence 0.65, however above this threshold, BWI would only achieve 0.8% recall, because the vast majority of its correct guesses are below a confidence threshold of 0.35. The highest F1 in this case comes from a confidence threshold of 0, with a precision of 52% and a recall of 24% (these numbers are taken from an individual train/test fold and thus differ slightly from the averages presented in Table 1). In other words, there is no way to improve performance by setting a confidence threshold, because there is not a smooth transition to a higher density of correct guesses as match confidence increases.

These analysis techniques immediately suggest a direct way of calibrating scores into probabilities—bin BWI's guesses by confidence threshold, and set the probability as the fraction of correct guesses in the bin. This essentially says that if for a given confidence threshold, say 75% of the guesses turn out to be correct, then a new match with the same confidence also has a 75% of being correct. This also has the desirable property that the more consistent the confidence scores, the more these calibrated probabilities become true probabilities. However, even for difficult tasks in which consistency is impossible, and thus there will not be a smooth increase in probability as confidence increases, these calibrated probabilities are still meaningful, because they will capture the remaining uncertainty in any guesses made.

9 Concluding remarks

Current information extraction methods are able to find and exploit regularities in text that come from local word ordering, punctuation usage, and distinctive word-level features like capitalization. This information is often sufficient for partially and highly structured documents because the regularity is at the surface level—in the use and ordering of the words themselves. However, these methods perform considerably worse when dealing with natural text, because such regularities are less apparent. Nevertheless, there are still important regularities in natural text documents, at the grammatical and semantic levels. We have shown that by revealing even limited grammatical information via XML results in considerably higher precision and recall for these types of tasks.

Despite the differences in behavior of these algorithms on document collections of varying regularity, there has been limited analysis of the specific relationship between regularity and performance. We proposed the SWI-Ratio as a quantitative measure of document regularity, and we have used it to illustrate this relationship in greater detail than has previously been possible. Since the SWI-Ratio objectively measures the relative regularity of a document collection (as the number of iterations SWI requires to perfectly cover all and only positive examples divided by the total number of positive examples), it is suitable for comparison across data sets of varying sizes and content.

While all the algorithms we studied perform worse on less regular document collections, they still exhibit consistent differences. Many current rule-based IE methods (including SWI, proposed in this paper) employ some form of set covering to combine multiple extraction rules. These methods are relatively simple to implement, however, they are all fundamentally limited in that they remove positive examples once covered, and cannot learn more rules than it takes to cover the entire training set. Boosting overcomes this problem by reweighting covered examples instead of removing them. We have shown that BWI exploits this property to learn additional useful rules even after all examples have been covered, and consistently outperforms SWI and related methods. Reweighting also helps by focusing BWI on learning specific rules for the exceptional cases missed by the general rules, resulting in higher precision. The combination of being able to learn accurate rules, and keep training to broaden coverage is the source of BWI's success.

While we have focused on rule-based IE methods as a class, and on BWI in particular, many of the observations made in this paper hold for information extraction as a whole. For example, while Hidden Markov Models employ an ostensibly different representation, they too tend to learn local flat regularities in adjacent word location, and distinctive use of punctuation, capitalization, and other surface regularities also exploited by BWI. Thus the discussion of what sources of information are currently being ignored are relevant to both classes of methods. We believe that there are many opportunities for improving both the performance and usefulness of current information extraction methods. We have outlined several suggestions for addressing the limitations of current methods discussed in this paper, all of which would benefit from further investigation.

Acknowledgements

The authors would like to thank Dayne Freitag for his input and for making the BWI code available, Mark Craven for giving us the natural text MEDLINE documents with annotated phrase segments, and MedExpert International, Inc. for its financial support of this research. This work was conducted as part of the California Institute for Telecommunications and Information Technology, Cal-(IT)^2.

References

Baluja, S., Mittal, V., & Sukthankar, R. (1999). Applying machine learning for high performance named-entity extraction. In *Proceedings of the Conference of the Pacific Association for Computational Linguistics* (pp. 365-378).

Bikel, D., Miller, S., Schwartz, R., & Weischedel, R. (1997). Nymble: a High-Performance Learning Name-finder. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pp. 194-201.

Califf, M. E., & Mooney, R. J. (1999). In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, Orlando, FL, pp. 328-334.

Califf, M. (1998). Rational Learning Techniques for Natural Language Information Extraction, Artificial Intelligence 1998, pg. 276.

Cardie, C. (2001). Rule Induction and Natural Language Applications of Rule Induction, http://www.cs.cornell.edu/Info/People/cardie/tutorial/tutorial.html.

Charniak, E. (2001). Immediate-Head Parsing for Language Models. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (2001).*

Ciravegna, F. (2001). (LP)^2, an Adaptive Algorithm for Information Extraction from Web-related Texts. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-2001).*

Clark, P. and Niblett, T. (1989). The CN2 Induction Algorithm. Machine Learning 3, pg. 261-283.

Cohen, W.W., & Singer, Y. (1999). A Simple, Fast, and Effective Rule Learner. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, 1999.

Eliassi-Rad, T., & Shavlik, J. (2001). A Theory-Refinement Approach to Information Extraction. In *Proceedings of the 18th International Conference on Machine Learning (ICML-2001).*

Freitag, D. & Kushmerick, N. (2000). Boosted Wrapper Induction, American Association for Artificial Intelligence 2000, pg. 577-583.

Kushmerick, N. (2000). Wrapper Induction: Efficiency and Expressiveness, Artificial Intelligence 2000, pg. 118.

Liu, L., Pu, C., & Ilan, W. (2000). XWrap: An XML-enabled Wrapper Construction System for Web Information Sources. In *Proceedings of the International Conference on Data Engineering, 2000.*

Margineantu, D. D., & Diettrich, T. G. (1997). Pruning adaptive boosting. In *Machine Learning: Proceedings of the Fourteenth International Conference*, pp. 211-218.

National Library of Medicine. (2001). The MEDLINE database, 2001. http://www.ncbi.nlm.gov/Pubmed/.

Michalski, S. (1980). Pattern recognition as rule-guided inductive inference. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2, 349-361.

Miller, G. (1995). Wordnet: A lexical database for English. *Communications of the ACM* 38(11):39-41

Muslea, I, Minton, S. & Knoblock, C. (1999). A Hierarchical Approach to Wrapper Induction.

Muslea, I. (1999). Extraction Patterns for Information Extraction: A Survey. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, 1999.

Quinlan, J.R. (1990). Learning logical definitions from Relations. Machine Learning, 5:239-266, 1990.

Ray, S. & Craven, M. (2001). Representing Sentence Structure in Hidden Markov Models for Information Extraction. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-2001).*

Riloff, E. (1996). Automatically Generating Extraction Patterns from Untagged Text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pp. 1044-1049.

Shapire, R. E. (1999). A Brief Introduction to Boosting. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-1999).*

Shinnou, H. (2001). Detection of errors in training data by using a decision list and AdaBoost. In *IJCAI-2001 Workshop, "Text Learning: Beyond Supervision"*, pp.61-65.

Yarowsky, D. (1994). Decision Lists for Lexical Ambiguity Resolution: Application to Accent Restoration in Spanish And French. In *Proceedings of the ACL '94*, pp. 77-95.

Yih, W-t. (1997). Template-based Information Extraction from Tree-structured HTML Documents. Masters thesis, National Taiwan University.

Data Set	SWI-Ratio	BWI			Fixed-BWI			Root-SWI			Greedy-SWI		
		Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
Latime-cc	0.013	0.996	1.000	0.998	1.000	0.985	0.993	1.000	0.975	0.986	1.000	0.948	0.973
Zagats-addr	0.011	1.000	0.937	0.967	1.000	0.549	0.703	1.000	0.549	0.703	1.000	0.575	0.724
QS-date	0.056	1.000	1.000	1.000	1.000	0.744	0.847	1.000	0.744	0.847	1.000	0.783	0.875
AbstractText	0.149	0.993	0.954	0.973	0.966	0.495	0.654	0.846	0.585	0.685	0.847	0.317	0.447
SA-speaker	0.246	0.791	0.592	0.677	0.887	0.446	0.586	0.777	0.457	0.565	0.904	0.342	0.494
SA-location	0.157	0.854	0.696	0.767	0.927	0.733	0.818	0.800	0.766	0.780	0.924	0.647	0.759
SA-stime	0.098	0.996	0.996	0.996	0.991	0.949	0.969	0.975	0.952	0.964	0.979	0.842	0.902
SA-etime	0.077	0.944	0.949	0.939	0.993	0.818	0.892	0.912	0.793	0.843	0.987	0.813	0.885
Jobs-id	0.068	1.000	1.000	1.000	1.000	0.956	0.978	1.000	0.956	0.978	0.996	0.829	0.902
Jobs-company	0.246	0.884	0.701	0.782	0.955	0.733	0.824	0.794	0.838	0.784	0.904	0.751	0.802
Jobs-title	0.381	0.596	0.432	0.501	0.661	0.479	0.547	0.480	0.658	0.546	0.660	0.477	0.549
YPD-protein	0.651	0.567	0.239	0.335	0.590	0.219	0.319	0.516	0.154	0.228	0.594	0.134	0.218
YPD-location	0.478	0.738	0.446	0.555	0.775	0.418	0.542	0.633	0.246	0.347	0.774	0.240	0.365
OMIM-gene	0.534	0.655	0.368	0.470	0.826	0.480	0.606	0.469	0.249	0.324	0.646	0.199	0.304
OMIM-disease	0.493	0.707	0.428	0.532	0.741	0.411	0.528	0.487	0.251	0.327	0.785	0.241	0.369

Table 1: SWI-Ratio and performance of the four IE methods examined on the 15 extraction tasks.

	cc	addr	date	abstract	speaker	loc.	stime	etime	id	comp.	title	protein	location	gene	disease
BWI	50	50	50	500	500	500	500	500	500	500	500	500	500	500	500
SWI	5.1	1.3	1.0	51.8	139.4	75.6	72.1	25.0	15.2	46.9	132.1	333.1	235.0	357.0	280.1
Root-SWI	3.6	1.3	1.0	48.4	110.7	66.8	62.0	17.9	1.0	46.3	119.6	322.5	229.7	344.6	278.8

Table 2: Number of boosting iterations used BWI, SWI, and Root-SWI on the 15 data sets.