Journal of Machine Learning Research 5 (2004) 499-527 Submitted 3/02; Revised 3/03; Published 5/04

Sources of Success for Boosted Wrapper Induction

David Kauchak DKAUCHAK@CS.UCSD.EDU
Department of Computer Science and Engineering
University of California at San Diego
La Jolla, CA 92093-0144, USA

Joseph Smarr JSMARR@STANFORD.EDU
Symbolic Systems Program
Stanford University
Stanford, CA 94305-2181, USA

Charles Elkan ELKAN@CS.UCSD.EDU
Department of Computer Science and Engineering
University of California at San Diego
La Jolla, CA 92093-0144, USA

Editor: Jaz Kandola

Abstract

In this paper, we examine an important recent rule-based information extraction (IE) technique named Boosted Wrapper Induction (BWI) by conducting experiments on a wider variety of tasks than previously studied, including tasks using several collections of natural text documents. We investigate systematically how each algorithmic component of BWI, in particular boosting, contributes to its success. We show that the benefit of boosting arises 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). As a quantitative indicator of the regularity of an extraction task, we propose a new measure that we call the SWI ratio. We show that this measure is a good predictor of IE success and a useful tool for analyzing IE tasks. Based on these results, we analyze the strengths and limitations of BWI. Specifically, we explain limitations in the information made available, and in the representations used. We also investigate the consequences of the fact that confidence values returned during extraction are not true probabilities. Next, we investigate the benefits of including grammatical and semantic information for natural text documents, as well as parse tree and attribute-value information for XML and HTML documents. We show experimentally that incorporating even limited grammatical information can increase the regularity of natural text extraction tasks, resulting in improved performance. We conclude with proposals for enriching the representational power of BWI and other IE methods to exploit these and other types of regularities.

Keywords: Boosted wrapper induction, information extraction, boosting

1. Introduction

Computerized text is abundant. Thousands of new web pages appear everyday. News, magazine, and journal articles are constantly being created online. Email has become one of the most popular ways of communicating. All these trends result in an enormous amount of text available in digital form, but these repositories of text are mostly untapped resources of information, and identifying specific desired information in them is a difficult task. Information extraction (IE) is the task of extracting relevant fragments of text from larger documents, to allow the fragments to be processed further in some automated way, for example to answer a user query. Examples of IE tasks include identifying the speaker featured in a talk announcement, finding the proteins mentioned in a biomedical journal article, and extracting the names 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. Successful techniques include statistical methods such as n-gram models, hidden Markov models, probabilistic context-free grammars (Califf, 1998), and rule-based methods that employ some form of machine learning. Rule-based methods have been especially popular in recent years. They use a variety of different approaches, but all recognize a number of common key facts. First, creating rules by hand is difficult and time-consuming (Riloff, 1996). For this reason, most systems generate their rules automatically given labeled or partially labeled data. Second, generating a single, general rule for extracting all instances of a given field is often impossible (Muslea et al., 1999). Therefore, most systems attempt to learn a number of rules that together cover the training examples for a field, and then combine these rules in some way.

Some recent techniques for generating rules in the realm of text extraction are called "wrapper induction" methods. These techniques have proved to be fairly successful for IE tasks in their intended domains, which are collections of highly structured documents such as web pages generated from a template script (Muslea et al., 1999; Kushmerick, 2000). However, wrapper induction methods do not extend well to natural language documents because of the specificity of the induced rules.

Recent research on improving the accuracy of weak classifiers using boosting (Schapire, 1999) has led to methods for learning classifiers that focus on examples that previous classifiers have found difficult. The AdaBoost algorithm works by iteratively learning a new classifier using a base learner and then reweighting the training examples to emphasize those that the current set of classifiers does not perform well on. The classifiers learned are then combined by voting with weights, where the weight of each classifier depends on its accuracy on its reweighted training set. Boosting has been shown theoretically to converge to an optimal combined classifier on the training set, and typically generalizes well in practice.

Boosted Wrapper Induction (BWI) is an IE technique that uses the AdaBoost algorithm to generate a general extraction procedure that combines a set of specific wrappers (Freitag and Kushmerick, 2000). Its authors showed that BWI performs well on a variety of tasks with moderately structured and highly structured documents. They compared BWI to several other IE methods, but exactly how boosting contributes to the success of BWI has not been investigated previously. Furthermore, BWI has been proposed as an IE method for unstructured natural language documents, but little has been done to examine its performance in this challenging direction.

In this paper, we investigate the benefit of boosting in BWI and also the performance of BWI on natural text documents. We compare the use of boosting in BWI with the most common approach to combining individual extraction rules: sequential covering. With sequential covering, extraction rules are ordered in some way according to performance on the training set. The best rule is chosen, and all examples that this rule correctly classifies are removed from the training set. A new best rule is learned, and the process is repeated until the entire training set has been covered. For a more detailed description of sequential covering see Cardie and Mooney (1998). Sequential covering has been used in some modern IE systems (Muslea et al., 1999) and is a common algorithmic idea in a wide range of other learning methods (Clark and Niblett, 1989; Michalski, 1980; Quinlan, 1990). Sequential covering is popular because it is simple to implement, tends to generate understandable and intuitive rules, and has achieved good performance.

This paper is divided into a number of sections. In Section 2, we briefly describe BWI and related techniques, providing a formalization of the problem and 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 the experiments in detail, with specific emphasis on how boosting affects the performance of BWI, and how performance relates to the regularity of the extraction task. In Section 5, we introduce the SWI ratio as a measure of task regularity, and further examine the connection between regularity and performance. In Section 6, we broaden the discussion and investigate important limitations of BWI and other current IE methods, focusing on what information is used and how it is represented, how results are scored and selected, and the efficiency of training and testing. Finally, in Section 7 we suggest ways to address these limitations, and provide experimental results that show that including grammatical information in the extraction process can increase exploitable regularity and therefore performance.

2. Overview of Algorithms and Terminology

In this section we present a brief review of the BWI approach to information extraction, including a precise problem description, a summary of 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 Information Extraction as a Classification Task

Most of the material in this section is based on Freitag and Kushmerick (2000). We present an abridged version here for convenience, starting with a review of relevant vocabulary and assumptions. A document is 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 subsequences of tokens matching a certain pattern from test documents. To learn the pattern, we reformulate the IE problem as a classification problem. Instead of thinking of the objective as a substring 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 a notion that arises from the parsing of the text into tokens. We want to learn two classifiers, that is two functions from boundaries 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 also used, for example, by Shinnou (2001) to find word boundaries in Japanese text, since written Japanese does not use spaces.

```
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)
     <p,s> -> FindBestDetector()
     add <p,s> to d
     remove positive examples covered by <p,s> from Y
     <p,s> = []

   return d
```

Figure 1: The SWI and GenerateDetectors algorithms.

Each of the two classifiers is represented as a set of boundary detectors, also called 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.⟩ matches "Who: Dr. John Smith" at the boundary between the colon ':' and the 'Dr'. Given beginning and ending classifiers, extraction is performed by identifying the beginning and end of a field and taking the tokens between the two points.

BWI separately learns two sets of boundary detectors for recognizing the alternative beginnings and endings of a target field to extract. During 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 do not match well. Once this process has been repeated for a fixed number of iterations, BWI returns the two sets of learned detectors, called fore detectors, F, and aft detectors, A, as well as a histogram, H, over the lengths (number of tokens) of the target field. The next subsection explains in more detail the BWI algorithm, with pseudocode, and identifies the differences between BWI and SWI.

2.2 Sequential Covering Wrapper Induction (SWI)

SWI uses the same framework as BWI with some modifications to the choice of individual detectors. The basic algorithm for SWI can be seen in Figure 1. SWI generates F and A independently by calling the GENERATEDETECTORS procedure, which is a sequential covering rule learner. During each iteration through the while loop, a new boundary detector is learned by the FINDBESTDETECTOR method. This method works by separately learning a prefix and suffix portion. As with BWI, this is done by starting with an empty sequence. Then, a single token is added by exhaustively enumerating all possible extensions of length L, picking the best one based on the scoring function (scoring functions will be discussed below) and adding the first token of that extension. This process is repeated, adding one token at a time, until the detector being generated no longer improves, as measured by the scoring function.

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 positive examples covered by the new rule are removed, leaving only uncovered examples. As soon as the set of detectors covers all positive training examples, this set is returned.

Given the sets of detectors F and A and the field length histogram, a field is extracted by applying the detectors in F and A to each boundary in the test document. For each boundary, an F score and an A score is computed as the sum of the confidence values of the detectors that match the boundary. A field is extracted if the product of the F score at the beginning of the field, the A score at the end of the field and the length histogram probability is greater than a predefined threshold (usually zero).

To clarify, there is a distinction between the actual text, which is a sequence of tokens, and the two functions to be learned, which indicate whether a boundary is the start or end of a field to be extracted. When the SWI algorithm "removes" the boundaries that are matched by a detector, these boundaries are relabeled in the training documents, but the actual sequence of tokens is not changed.

There are two key differences between BWI and SWI. First, as mentioned above, with sequential covering covered positive training examples are removed. In BWI, examples are reweighted but never relabeled. Second, the scoring functions for the extensions and detectors are slightly different. We investigate two different versions of SWI. The basic SWI algorithm uses a simple greedy method. The score of an extension or detector is the number of positive examples that it covers if it covers no negative examples, zero otherwise. 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, 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 and Singer, 1999):

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

Notice that for Root-SWI the two sums are the numbers of examples covered because 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 particularly important difference between the two different methods.

2.3 Wildcards

One additional aspect of both BWI and SWI that needs further explanation is the use of wildcards. Both BWI and SWI extend boundary detectors using tokens as found in the training examples. However, using literal tokens does not allow these algorithms to recognize common regularities in the documents. For example, given the task of identifying phone numbers in a document, the algorithms as described above would be unable to match a future phone numbers unless that specific number had already been seen in a training example. To overcome this limitation, wildcards that represent categories of tokens are used. Freitag and Kushmerick describe eight basic wildcards, which are also used in our work, representing alphabetic tokens, alphanumeric tokens, tokens beginning with a capitalized letter, tokens beginning with a lower-case letter, single character tokens, numeric tokens, punctuation tokens, and any token.

3. Experiments and Results

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

3.1 Document Collections and IE Tasks

In order to compare BWI, Greedy-SWI and Root-SWI we examine the three algorithms on 15 different information extraction tasks using 8 different document collections. Many of these tasks are standard and have been used in testing a variety of IE techniques. The document collections can be categorized into three groups: natural (unstructured) text, partially structured text, and highly structured text. The breadth of these collections allows for a comparison of the algorithms on a wider spectrum of tasks than has previously been studied. The natural text documents are biomedical journal abstracts and contain little obvious formatting information. The partially structured documents resemble natural text, but also contain document-level structure and have standardized formatting. These documents consist primarily of natural language, but contain 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 are web pages that have been automatically generated. They contain reliable regularities, including location of content, punctuation and style, and non-natural language formatting such as HTML tags. The partially structured and highly structured document collections were obtained primarily from RISE (1998) while the natural text documents were obtained from the National Library of Medicine MEDLINE abstract database (National Library of Medicine, 2001).

We use two collections of natural language documents, both consisting of individual sentences from MEDLINE abstracts. The first collection consists of 555 sentences tagged with the names of proteins and their subcellular locations. These sentences were automatically selected and tagged from larger documents by searching for known proteins and locations contained in the Yeast-Protein-Database (*YPD*). Because of the automated labeling procedure, the labels are incomplete and sometimes inaccurate. Ray and Craven (2001) estimate an error rate in labeling between 10% and 15%. 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 and Craven (2001) and Eliassi-Rad and Shavlik (2001).

It should be noted that the original *OMIM* and *YPD* document collections contain many sentences labeled with no training examples of fields to extract, because the collections were created for extracting field pairs only. Therefore, only in cases where both fields in a pair were found was any training label added. Because BWI is designed to extract only single fields, we remove these sentences. Many of them in fact contain examples of desired fields, so when unlabeled, they present incorrect training information. Moreover, the unlabeled documents originally outnumber the labeled documents by almost an order of magnitude. Our restricted sentence collections still pose challenging information extraction tasks, but performance on these collections cannot be compared to performance reported for extracting field pairs in the original collections.

For the partially structured extraction tasks, we use three different document collections. The first two are taken from RISE, and consist of 486 speaker announcements (*SA*) and 298 Usenet job announcements (*Jobs*). In the speaker announcement collection, four different fields are 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 are extracted: the message identifier code (*Jobs-id*), the name of the company (*Jobs-company*) and the title of the available position (*Jobs-title*).

We created the third collection of partially structured documents from 630 MEDLINE article citations. These documents contain full MEDLINE bibliographic data for articles on hip arthroplasty surgery, i.e. author, journal, and publication information, 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, if the citation includes one (*AbstractText*). The abstracts are generally large bodies of text, but without any consistent beginning or ending marker, and the information that precedes or follows the abstract text 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 the Zagat Guide to Los Angeles Restaurants (*Zagat*), where the task is to extract the restaurant's address (*Zagat-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*). Although these collections contain a small number of documents, the individual documents typically comprise several separate entries, i.e. there are often multiple stock quote responses or restaurant reviews per web page.

3.2 Experimental Design

The performance of the different rule-based IE methods on the tasks described above is measured using three different standard metrics: precision, recall, and F1, the harmonic mean of precision and recall. F1 is used as the principal metric to compare different algorithms for several reasons: it balances precision and recall, it is common in previous research on information extraction and information retrieval, and it allows experimental results to be quantified with a single number.

Given that we analyze BWI, SWI and related algorithms in a range of applications, and in many domains both precision and recall are important, we consider the F1 measure to be better than alternative single measures, but it is still somewhat arbitrary. For many applications, there are asymmetric costs associated with false positives and false negatives. For example, high precision is necessary when IE is used to populate a database automatically, while high recall is important when IE is used to find descriptions of rare events that are later reviewed by a human analyst. In these applications it is beneficial to make explicit the relative costs of false positives and false negatives, and to use a cost-sensitive learning method (Elkan, 2001).

Each result presented is the average of ten random 75% train/25% test splits. For all algorithms, a lookahead value, L, of 3 tokens is used. For the natural and partially structured texts (*YPD*, *OMIM*, *SA*, *Jobs*, and *AbstractText*), the default BWI wildcard set is used (the eight wildcards described in Section 2.3), and for the web pages (*LATimes*, *Zagat*, and *QS*), three lexical wildcards are used in addition; these represent last names, first names, and words not found in the standard Unix list

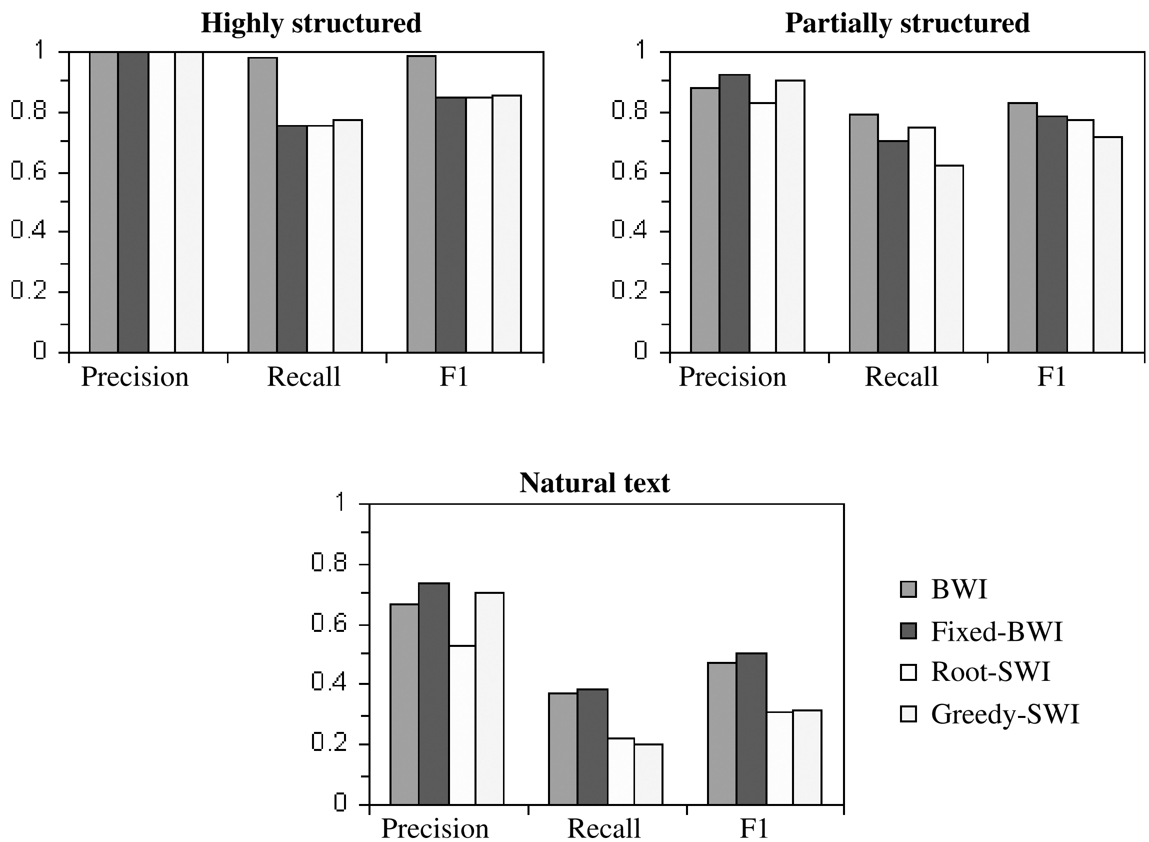

Figure 2: Performance of the four IE methods examined in this paper, separated by overall regularity of document collection (indicated above each chart), averaged over all tasks within the given collections.

of words /USR/DICT/WORDS. 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.

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 used by Freitag and Kushmerick (2000): for the *YPD*, *OMIM*, *SA*, *Jobs*, and *AbstractText* document collections, 500 rounds of boosting were used; for the *LATimes*, *Zagat*, 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 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: 323 rounds for *YPD-protein*, one round for *QS-date*, and so on; this method is called Fixed-BWI.

3.3 Experimental Results

To investigate whether BWI outperforms the set-covering approach of SWI, we compare Greedy-SWI and BWI. BWI appears to perform better, since 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 partially 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 next logical step is determining what part of the success of BWI 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 compare 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 partially structured tasks, despite the fact that they use an identical scoring function.

There are only two differences between BWI and Root-SWI. First, after each iteration Root-SWI removes all positive examples covered, 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. Table 2 shows that Root-SWI always terminates after many fewer iterations than were set for BWI. Freitag and Kushmerick (2000) examined the performance of BWI as the number of rounds of boosting increases, and found improvement in many cases even after more than 500 iterations.

To investigate this issue further, we ran BWI for the same number of iterations as it takes 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 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 partially structured tasks, Fixed-BWI tends to exhibit higher precision but lower recall than Root-SWI, resulting in similar F1 values. In the unstructured tasks, Fixed-BWI has considerably higher precision and recall than Root-SWI.

3.4 Statistical Significance of Experiments

In this section and in following sections (4 and 7) we compare the performance of algorithmic variations on the different data sets. When comparing the performance of algorithms, particularly with limited data, it is common to verify that the differences in performance are not due to chance. This is usually done using a statistical test of significance. We checked the validity of all of our comparisons using the popular paired *t*-test. All comparisons we examined are statistically significant based on this test. The worst significant level found was 0.0005 and most comparisons were significant with much greater confidence. We chose the *t*-test because it has been commonly used in the past. Dietterich (1998) mentions a number of concerns regarding the *t*-test, however, the ten 75%/25% splits lessens these problems and all the tests examined were much more confident than the common 0.01 significance level.

4. Analysis of experimental results

In this section, we discuss our experimental results from two different angles. We examine the effect of the algorithmic differences between 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 experiments performed yield a detailed understanding of how the algorithmic differences between BWI and SWI allow BWI to achieve consistently higher F1 values than SWI. The most important difference 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 the fact 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 does not permit any negative examples to be covered by any rules. This results in lower recall than with the other three methods, because general rules with wide coverage are hard to learn 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. Unlike BWI, SWI is 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. (Usually many fewer detectors are learned because multiple examples are covered by single rules.) In contrast, after each iteration BWI reweights examples, but does not remove them entirely, so there is no limit in principle to how many rules can be learned. 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 The Consequences of Task Difficulty

As just explained, boosting gives BWI the consistent advantage in F1 performance observed in the experiments of Section 3. However, the difficulty of the extraction task also has a pronounced effect on the performance of different IE methods. We now turn to a specific investigation of each document collection.

4.2.1 Highly Structured IE Tasks

These tasks are the easiest for all methods. It is no coincidence that they are often called "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, SWI does not achieve near-perfect recall. While these tasks are highly regular, the regularities learned on the first couple of iterations are not the only important ones. Neither changing the scoring function from "greedy" to "root" nor changing the iteration method from set covering to boosting solves 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 cases that are missed by a smaller set of rules. Thus BWI achieves higher recall than the other methods on these extraction tasks.

4.2.2 PARTIALLY STRUCTURED IE TASKS

For these tasks, we see the largest variation in performance between the IE algorithms investigated. Compared to Greedy-SWI, Root-SWI has increased recall, decreased precision, and similar F1. By allowing rules to cover some negative training examples, more general rules can be learned that have higher recall, but also cover some negative test examples, causing 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 results in a precision/recall tradeoff with little change in F1. As mentioned earlier, boosting reweights the examples to focus on the hard cases. This results in rules that are very specific, with higher precision but also lower recall. Boosting also results in a slower learning curve, measured in performance versus number of rules learned, because positive examples are often covered multiple times before all examples have been covered once. SWI removes all covered examples, focusing at each iteration on a new set of examples. The consequence is that Fixed-BWI cannot 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 enough iterations of boosting are used, BWI compensates for its slow start by learning enough rules to ensure high recall without sacrificing high precision.

4.2.3 UNSTRUCTURED TEXT IE TASKS

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, on the natural text tasks all the IE methods investigated have relatively low precision, despite the bias towards high precision of the boundary detectors. All the methods also have relatively low recall. This is mainly due to the fact that the algorithms often learn little more than the specific examples seen during training, which usually do not 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: the fore detectors have no prefix, and the target field as the suffix, and vice versa for the aft detectors. Changing the SWI scoring function from "greedy" to "root" results in a marked decrease in precision with only a small increase in recall. Even when negative examples may be covered, the rules learned are not sufficiently general to increase recall significantly.

Unlike on the partially and highly structured IE tasks, Fixed-BWI has both higher precision and higher recall than Root-SWI on the natural text tasks. Higher precision with boosting is explainable as above. Boosting also increases recall because while already covered examples are down-weighted during boosting, they are not removed entirely. Thus, BWI remains sensitive to the performance on all training examples of all the rules it learns. In contrast, SWI quickly captures the regularities that can be easily found, and then begins covering positive training examples one at a time, ignoring the performance of new rules on already covered positive examples. The problem that SWI learns many rules with low recall is compounded by the fact that it never removes negative examples. This behavior of SWI is particularly troublesome because the labelings of the collections of natural text documents are partially inaccurate, as explained in Section 3.1. Unlike on the more structured tasks, allowing BWI to run for 500 iterations causes 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 more iterations on natural text tasks than on easier tasks. The 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. All the algorithms we consider easily handle highly structured documents such as those generated from a database and an HTML template, while natural text documents are uniformly more difficult. It is thus interesting to investigate methods for measuring the regularity of an information extraction task numerically and objectively, as opposed to subjectively and qualitatively. In this section we provide such a measure, which we find to be useful in our work and which we expect to be useful in future research.

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 number of iterations SWI takes to cover all the positive examples in a training set, divided by the total number of positive examples in the training set. We call this measure the *SWI ratio*. In the most regular limit, a single rule would cover an infinite number of documents, and thus the SWI ratio would be $1/\infty = 0$. At the other extreme, in a completely 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 $N/N = 1$. 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. So, the SWI ratio will always be between 0 and 1, with a lower value indicating a more regular extraction task.

The SWI ratio is a sensible and well-motivated measure of task regularity for several reasons. First, it is a relative measure with respect to the number of training examples, so one can use it to compare the regularity of tasks with small and large training collections of documents. Second, it is simple and objective, because SWI is a straightforward algorithm, with no parameters to set other than the lookahead for extending boundary-detector rules, which is fixed at $L = 3$ in all our tests, and the set of wildcards, for which using the default set of eight makes sense in a wide range of domains. Third, the SWI ratio is efficient to compute, and it is practical to run SWI before or while running any other technique. Finally, and most importantly, the SWI ratio correlates with task regularity as perceived intuitively by humans. Table 1 shows the SWI ratios for all of the extraction tasks. The highly structured tasks have an average SWI ratio of 0.027, the partially structured tasks have an average SWI ratio of 0.178 and the unstructured tasks have an average SWI ratio of 0.539. The tasks that are perceived by humans as being the most regular, such as dates, times, credit card numbers and addresses, have low SWI ratio values. Rules can easily be generated by hand to extract these

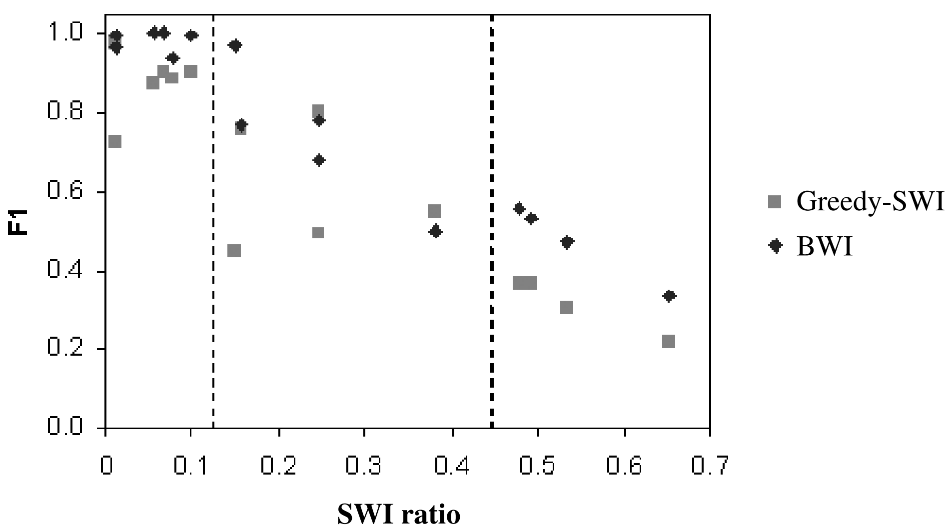

Figure 3: F1 performance for BWI and Greedy-SWI on the 15 different extraction tasks, plotted versus the SWI ratio of the task, with separations between highly structured, partially structured, and natural text.

fields. On the other hand, the tasks involving the MEDLINE articles, which are human-generated documents, have the highest SWI ratio values.

The SWI ratio has a similar motivation to other measures of regularity. Rendell and Cho (1990) examine the concentration of the data set, which is the distribution of the positive instances throughout the instance space. They introduce the idea of a peak (a neighborhood of values that has a class membership that is greater than average) and show that as the number of peaks increases (i.e. the concentration of the data decreases) the performance decreases. The SWI ratio can be seen as a similar, though not identical, measure of concentration. The SWI ratio is also similar to a VC bound on generalization error. Intuitively, it is the capacity (i.e. complexity) of the space of classifiers needed to achieve zero training error divided by the number of training examples; this ratio is expected theoretically to be related to the generalization error rate (Vapnik et al., 1994).

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 class of tasks. Each value in Figure 2 is an average over several extraction tasks. Using the SWI ratio we can compare algorithms on a per-task basis. This reveals in greater detail both how performance is related to task difficulty, and whether BWI reliably outperforms SWI.

In Figure 3, F1 values for 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 F1 difference on those tasks is small. By plotting the performance of each algorithm versus the SWI ratio, we can examine how the two algorithms succeed as the regularity of the tasks decreases.

There is a consistent decline in performance as the tasks decrease in human-perceived regularity, for both algorithms. Also, there is no clear divergence in performance between the two algorithms as the regularity of the task decreases. So, although BWI appears to reliably outperform SWI, both algorithms are still limited by the regularity of the task to be solved.

5.3 The SWI Ratio as a Tool for Future Research

A numerical and objective measure of the regularity of information extraction tasks is useful not only for predicting how well a known IE method will do on a new task, but also for guiding research to improve IE methods. In Sections 5.1 and 5.2 above, the human perception of task regularity is confirmed by measurement with the SWI ratio, while in Section 7.1 below, the SWI ratio is used to show how the regularity of an IE task can be increased under certain conditions. We hope that future research will show how to use the SWI ratio to help in choosing the right technique for a given IE task. For example, a given IE method may tend to be particularly suitable for tasks whose SWI ratio falls in a particular range.

6. Opportunities for Improving BWI

Having investigated BWI as an important recent information extraction technique, and having understood its advantages over variant methods, we now broaden the discussion to investigate a number of limitations of BWI. In particular, we examine types of information that BWI fails to use, or does not represent completely, as well as speed and efficiency problems. We examine the limitations of BWI in the hope of creating a useful list of important issues to consider when designing and evaluating many future IE systems.

As shown in Section 3 above and in numerous other papers, many IE systems perform quite well on fairly structured tasks. If the target field to be extracted is canonically preceded or followed by a given set of tokens, or by tokens of a distinct type, represented by the wildcards available, boundary detectors easily represent this context. This is a common occurrence in highly structured and partially structured documents, where fields are often preceded by identifying labels (e.g. “Speaker: Dr. X”), or followed by identifiable pieces of information (e.g. in the Zagat survey, a 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” are common prefixes, and are learned early by BWI. In general, human authors often provide a given type of information only in a certain context, and specific lead-in words or following words are used repeatedly.

While rule-based IE methods are primarily designed to identify contexts outside target fields, BWI and other methods do also learn about some of the regularities that occur inside 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 can learn what the first few tokens of a target field look like, if the field tends to have a regular beginning, and the aft detectors can learn what the last few tokens look like.

With short fields, individual boundary detectors often memorize instances of the target field. This happens in the MEDLINE tasks, where specific gene names, protein names, etc. are memorized when their context is not 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 typical starting and ending tokens of a target field, BWI learns a probability distribution for the number of tokens in the field. Specifically, the field length is recorded for each training example, and this histogram is normalized into a probability distribution once training is complete. In BWI, length is the only piece of information that ties fore detectors and aft detectors together. Length information can be extremely useful. When BWI runs on a new test document, it first notes every fore and aft detector that fire, and then pairs them up to find specific token sequences. BWI only keeps a match that is consistent with the length distribution. Specifically, it drops matches whose fore and aft boundaries are farther apart or closer together than seen during training. Also, given two overlapping matches of equal detector confidence, it prefers 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 the different pattern types, to distinguish 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 so reflects the generality of the rule. In SWI, rules that cover the most examples are typically learned first, so there is a ranking of rules.

6.1 Representational Expressiveness

BWI learns sets of fore and aft boundary detectors and a histogram of the lengths of the fields. The boundary detectors are short sequences of specific words or wildcards that directly precede or follow the target field or that are found within the target field. As a consequence of this representation, there are valuable types of regularity that cannot be captured, including a detailed model of the field to be extracted, the global location of the field within documents, and structural information such as that exposed by a grammatical parse or an HTML/XML document tree.

6.1.1 LIMITED 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 section, BWI can learn the canonical beginnings and endings of a target field, as well as a distribution over the length of the 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 generalization. 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 dismissed, no matter how compelling the boundary detection. This does not cause problems for very short fields, when there are only a few possible lengths, but it is a major 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, many reasonable abstract lengths are never seen in training data.

In addition to 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. Often, more information about the canonical form of the target field could be exploited. There has been a great deal of work on modeling fragments of text to identify them in larger documents, most under the rubric Named Entity Extraction. For example, Baluja et al. (1999) reports impressive performance 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 effort in this field (Bikel et al., 1997). Combining these field-finding methods with the BWI context-finding methods should yield superior performance.

6.1.2 LIMITED EXPRESSIVENESS OF BOUNDARY DETECTORS

Boundary detectors are designed to capture the flat, nearby context of a field to be extracted, by learning short sequences of surrounding tokens. They are good at capturing regular, sequential information around the field to be extracted, resulting in high precision. However, current boundary detectors are not effective in partially structured and natural texts, where regularities in context are less consistent and reliable. In these domains, many detectors only cover one or a few examples, and collectively the detectors have low recall.

A second limitation of existing boundary detectors is that they cannot represent the grammatical structure of sentences, or most structural information present in HTML or XML documents. Boundary detectors can only capture information about the tokens that appear directly before or after a target field in the linear sequence of tokens in a document. BWI cannot represent or learn any information about the parent nodes, siblings, or child position in the grammar, or XML or HTML tree, to which target fields implicitly belong. In addition, the boundary detector representation 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 recognizing a field, global information about the relative position of the field within the entire document can also be important. For example, despite the impressive performance of BWI on the abstract detection task, 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 the existing BWI representations. Additional knowledge that BWI is unable to learn or use includes “the field is in the second sentence of the second paragraph if it is anywhere,” “the field is never more than one sentence long,” “the field has no line breaks inside it,” and so on.

6.2 Scoring of Candidate Matches

Each match that BWI returns when applied to a test document has an associated confidence score, which is the product of the confidences of the fore and aft boundary detectors, and the learned probability of the field length of the match. These scores are useful for identifying the most important detectors, and also for choosing a threshold to obtain a precision/recall tradeoff.

While the score of a candidate match is correlated with the chance that the proposed match is correct, these scores are not as useful as actual probabilities would be. First, the confidence scores are unbounded positive real numbers, so it is difficult to compare the scores of matches in different document collections, where the typical range of scores is different. For the same reason, it is hard to set absolute confidence thresholds for accepting proposed matches.

Second, it is 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 entity for analysis to improve performance. Finally, without true probabilities, it is difficult to use match scores in a larger framework, such as Bayesian evidence combination or decision-theoretic choice of action. The basic problem is that while confidence scores are useful for stating the relative merit 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 merit.

Another problem with confidence scores is that no distinction is made between a fragment of text that has appeared numerous times in the training set as a negative example, a fragment unlike any seen before, and a fragment where one boundary detector has confidently fired but the other has not. These all have confidence score zero, so there is no way to tell a "confident rejection" from a "near miss."

BWI also implicitly assumes that fields of just one type are extracted from a given document collection. 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 ignores the relative position of different fields. For example, in the speaker announcements domain, even though four fields are extracted from each document, the field extractors are all trained and tested independently, and the presence of one field is not used to predict the location of another. It is difficult to extend methods like BWI to extract relationships in text, a task that is often as important as extracting individual fields (Muslea et al., 1999).

6.3 Efficiency

In addition to precision and recall, the speed and efficiency of training and testing is also a critical issue in making IE systems practical for solving real-world problems.

6.3.1 BWI IS SLOW TO TRAIN AND TEST, AND NOT INCREMENTAL

Even on a modern workstation, training BWI on a single field with a few hundred documents can take several hours, and testing can take several minutes. The slowness of training and testing makes using larger document collections, or larger lookahead for detectors, prohibitive, even though both may be necessary to achieve high performance on complex tasks.

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, with a specified lookahead parameter, L, and repeated until no better rule can be found. Finding a boundary detector extension is exponential in L, because every combination of tokens and wildcards is enumerated and scored. So, even modest lookahead values are prohibitively expensive. A value of $L = 3$ is normally used to achieve a balance between efficiency and performance. Freitag and Kushmerick note that $L = 3$ is usually sufficient, but for some tasks a value up to $L = 8$ is required to achieve reasonable results.

Although testing is considerably faster than training, it too is inefficient. 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. This is a practice used by Cohen and Singer (1999) in SLIPPER, and by Ciravegna (2001) in $(LP)^2$ for partially structured tasks. It is doubtful how useful redundancy elimination would be for less regular document collections, which are the ones for which many detectors must be learned. There are also more sophisticated methods to compress a set of rules, many of which are somewhat lossy (Margineantu and Dietterich, 1997; Yarowsky, 1994). For these methods, the tradeoff between speed and accuracy would need to be measured.

After BWI has been trained on a given set of labeled documents, if 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 extra information from a few additional documents, even though the 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 performance on all training examples, so adding even a few extra documents can mean that training takes a very different direction.

6.3.2 BWI Cannot Determine when to Stop Boosting

While the ability of BWI to boost for a fixed number of rounds instead of terminating as soon as all positive examples are covered is clearly one of its advantages, it is hard to decide how many rounds to use. Depending on the difficulty of the task, a given number of rounds may be insufficient to learn all the cases, or it may lead to overfitting, or to redundancy. Ideally, BWI would continue boosting for as long as useful, and then stop when continuing to boost would do more harm than good.

Cohen and Singer (1999) address the problem of when to stop boosting directly in the design of SLIPPER by using internal five-fold validation on a held-out part of the training set. 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 leads to the lowest average error on the held-out set, finally training for that number of rounds on the full training set. This process is reasonable, and sensitive to the difficulty of the task, but it has several drawbacks: the process of training is made six times slower and the free parameter of the number of boosting rounds is replaced by another free parameter, the maximum number of rounds to try during cross-validation.

It may be possible to use the BWI 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 may be possible to set an absolute or relative threshold below which to stop boosting. A relative threshold would measure when the curve has flattened out, and new detectors are 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 too likely to propose spurious matches.

7. Extending Rule-Based IE Methods

In this section we present several suggestions for overcoming the deficiencies of BWI analyzed in the previous section. These suggestions are applicable to BWI and to alternative learning algorithms for IE. We concentrate on identifying new sources of information to consider, constructing new representations for handling them, and producing more meaningful output. In some cases, we present preliminary results that corroborate our hypotheses. In other cases, we cite existing work by other researchers that we believe represent steps in the right direction.

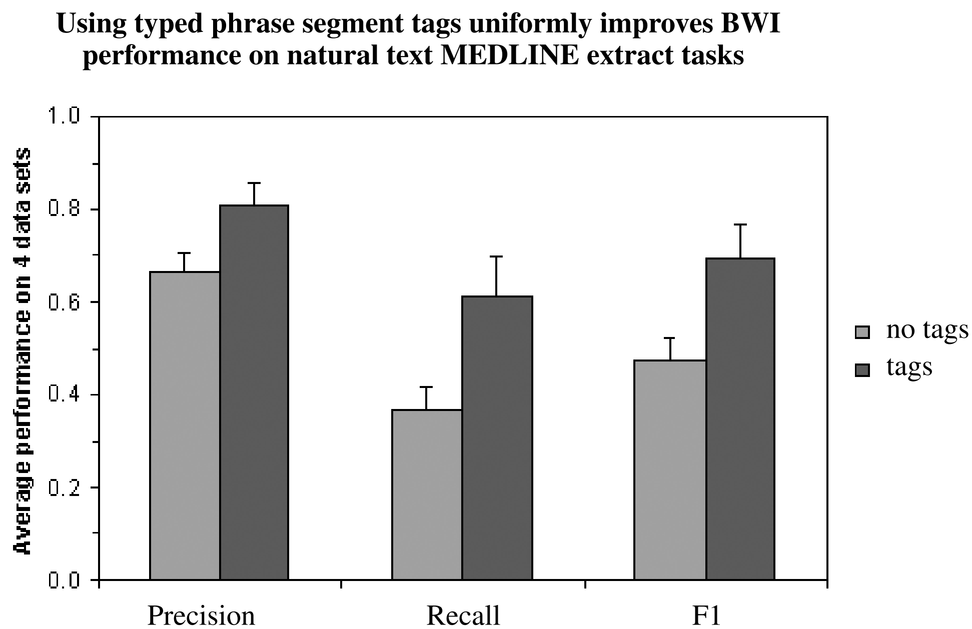

Figure 4: Performance of BWI averaged across the four natural text extraction tasks, with and without the use of typed phrase segments. Means are shown with standard error bars.

7.1 Exploiting the Grammatical Structure of Sentences

Section 6.1.2 discusses the importance of using grammatical structure in natural language or hierarchical structure in HTML and XML documents. Ray and Craven (2001) have taken an important first step in this direction by preprocessing natural text with a shallow parser, and then flattening the parser output by dividing sentences into typed phrase segments. The text is then marked up with this grammatical information, which is used as part of the information extraction process. Ray and Craven use hidden Markov models, but their 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 are not perfect, and the flattening often causes further distortion, this procedure is fairly consistent in labeling noun, verb, and prepositional phrases. An information extraction system can then learn rules that include 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 Section 3, which use the document collections of Ray and Craven without grammatical information, we also obtained results using phrase segment information. We used BWI to extract the four individual fields in the two relations that Ray and Craven

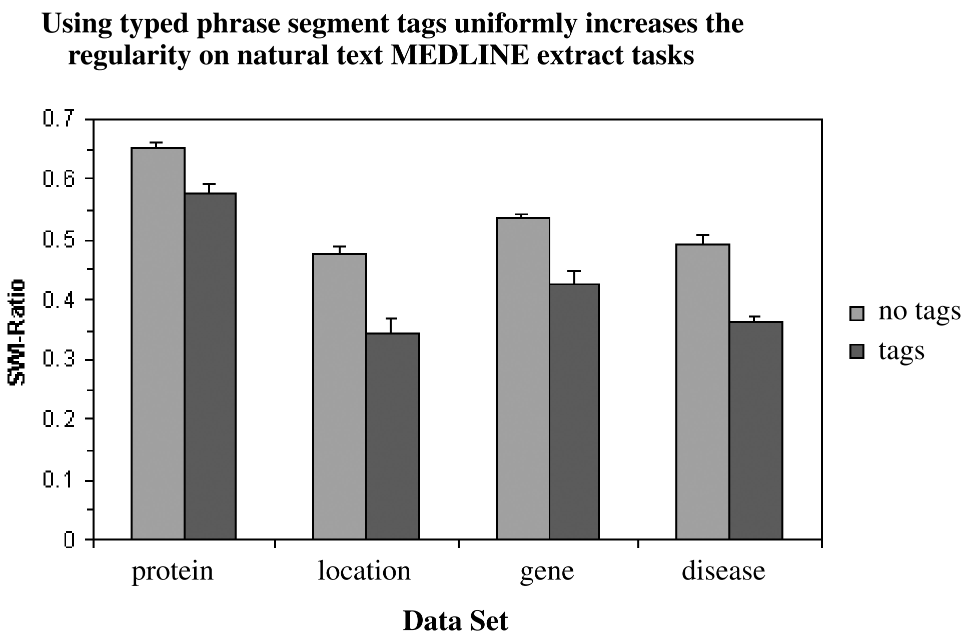

Figure 5: Average SWI ratio for the four natural extraction tasks, with and without the use of typed phrase segments. Means are shown with standard error bars.

study (proteins and their localizations, genes and their associated diseases). We ran identical tests with the grammatical tags inserted as single tokens on either side of a segment (a different tag is specified for the end and start), and without these tags. Including the tags uniformly and considerably improves both precision and recall, for all four extraction tasks (Figure 4). In fact, all four tasks see double-digit percentage increases in precision, recall, and F1, with average increases 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 it 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, since there are many non-protein noun phrases in MEDLINE articles.

We hypothesize that including typed phrase segment information actually regularizes the extraction task, enabling rules to gain greater positive coverage without increasing negative coverage. Using the SWI ratio described in Section 5, we can test this hypothesis quantitatively. Figure 5 shows the SWI ratios for all four extraction tasks with and without phrase segment tags. As expected, there are double-digit percentage decreases in the SWI ratio for all four tasks, with an average reduction of 21%. Recall that a lower SWI ratio indicates 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 grammatical information, even with existing rule-based IE methods, can be a considerable advantage for both recall and precision, and is worth investigating in more detail.

Our experiments show that grammatical information might be exploited even more. Given how the BWI tokenizer is implemented, all grammatical tags are encoded as capitalized neologisms such as LLLNPSEGMENT. Surprisingly, the rules learned by BWI use the "all upper-case" wildcard more often than they mention a specific grammatical tag. In other words, BWI learns rules of the form "start matching at the boundary of a phrasal chunk" more often than rules that specify a particular grammatical construct such as a noun phrase. An explanation for this behavior is that fragments of text to be extracted are usually well-formed syntactically, so "chunking" the text helps BWI focus on fragments of text that do not cross syntactic boundaries. This focus helps eliminate incorrect fragments and helps identify correct fragments, thus improving both precision and recall. This is similar to the benefit that information retrieval techniques achieve from segmenting documents into coherent regions.

In order to use grammatical information more, the existing formulation of boundary detectors may need to be changed. While inserting XML tags to indicate typed phrase segments is useful, this approach is unprincipled, as it loses the distinction between meta-level information and the text itself. The knowledge representation used by boundary detectors should be extended to express regularities in implicit higher-level information, as well as in explicit token information in a document. When using typed phrase segments, we can increase performance without changing the simple, flat representation currently used. However, this approach cannot make recursive structure explicit, so it can only be taken so far.

The success of including even limited grammatical information immediately raises the question of what additional grammatical information can be used, and how beneficial it might be. It is plausible that regularities in field context such as argument position in a verb phrase, subject/object distinction, and so on can be valuable. For example, Charniak (2001) has shown that probabilistically parsing sentences is greatly aided by conditioning on information about the linguistic head of the current phrase, even if the head is several tokens away in the flat representation. Charniak's finding is evidence that linguistic head information is an important source of regularity, which is exactly what rule-based IE methods are designed to exploit. There have also been several other attempts to exploit limited grammatical analysis for information extraction, including AutoSlog (Riloff, 1996), FASTUS (Appelt et al., 1995), and CRYSTAL (Soderland et al., 1995), all of which demonstrate the potential value of this type of information. As pointed out by Grishman (1997), improved performance is often achieved only when such information is used in a conservative way, for example looking only for specific grammatical patterns, or using grammatical information only in contexts that are unambiguous.

7.2 Binning Histograms of Field Length

Section 3 presents results using the field lengths seen in the training set when computing confidence values for possible extractions. When the variance of field lengths is small, this method works well. However, as mentioned in Section 6.1.1, some tasks involve a wide range of field lengths, for example the task of identifying abstracts in documents.

On the AbstractText task BWI gives an F1 of only 71%. The main reason for this poor performance is a recall of 62%. Because BWI gives a score of zero to any potential field with length never seen in the training set, many correct fields are excluded in test documents even though boundary detectors indicate that these fields should be extracted: relying on raw length counts from the training data causes 45% of the predictions made by boundary detectors on the test set to be dropped from consideration.

A simple response is to ignore the field lengths and extract fields based only upon the information provided by the boundary detectors. For the AbstractText task, this method increases F1 to 97.8%. However, this method requires a human to analyze the task in advance to decide whether lengths should be used. A more general solution would instead generalize automatically from the lengths seen in the training set.

The difficulty is that the length histogram does not generalize from the lengths seen in the training set. To generalize, the histogram bins must be expanded to allow for lengths never seen in the training set. We take a simple approach by using a fixed number of equal width bins.

We would like to make the bins start at the true minimum length and end at the true maximum. Unfortunately, we do not know the true minimum and maximum because we only have a sample of the data. The maximum likelihood estimates (MLEs) for these parameters are the minimum and maximum of the sample (i.e. the smallest and largest lengths seen in the training data). Unfortunately, the MLEs are biased: they tend to underestimate the maximum and overestimate the minimum, particularly for small sample sizes. Fortunately, we can compute unbiased estimates for the true max and min:

$$\text{max\_estimate}(x) = max(x)\frac{n+1}{n} \qquad\qquad \text{min\_estimate}(x) = min(x)\frac{n-1}{n},$$

where x is the sample of size n.

The unbiased estimates above are valid for a uniform distribution. For bell-shaped distributions, these estimates need to be expanded further beyond the sample minimum and maximum. For this reason, in our experiments we use ten bins of uniform width with 1/18th of the range beyond the sample max and min. In other words, we extend half a bin width beyond the sample minimum and maximum.

These bin sizes give F1 of 97.0% on the AbstractText task. This is slightly worse than the F1 achieved by ignoring length, but the binning method performs well on both highly irregular tasks such as the AbstractText task and more regular tasks like those examined in Section 3 and does not require a human to examine the method a priori.

7.3 Handling XML or HTML Structure and Information

Just as the grammatical structure of a sentence presents opportunities for exploiting structural regularities, the hierarchical structure and attribute information available in 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 using 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. to match a tag with the same name, to require specific attributes and/or values, etc.

Muslea et al. (1999) 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 it to take advantage of global landmarks for finding and extracting text fields.

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 are 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. This is an excellent first approach to parsing hierarchically arranged documents, but using Skip-To only captures some of the information available in a complete DOM representation.

A second approach to exploiting the structural information embedded in web pages and XML documents is due to Yih (1997), who 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, 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. Constructing templates may be less of a problem with XML documents, but only if their 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. (2000) with their "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.

7.4 Extending the Expressiveness of Boundary Detectors

One simple solution to the problem of exploiting more grammatical and structural information is the development of a more sophisticated set of wildcards. Currently, wildcards only represent word-level syntactic classes, such as "all upper-case", "begins with a lower-case letter", "contains only digits", and so on. While these are useful generalizations over matching individual tokens, they are also extremely broad. A potential middle ground consists of 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.).

Encouraging results in this direction are already available. For example, Ciravegna's $(LP)^2$ system uses word morphology and part-of-speech information to generalize the rules it initially learns (Ciravegna, 2001), a process similar to using lexical and part-of-speech wildcards. Ciravegna's results are comparable to those with other state of the art methods, including BWI. While $(LP)^2$ also does rule correction and rule pruning, Ciravegna says that "the use of NLP for generalization" is the most responsible for the performance of the system. Another clever use of generalizations can be found in RAPIER (Califf and Mooney, 1999), which uses WordNet (Miller, 1995) to find hypernyms (a semantic class of words to which the target word belongs), then uses these in its learned extraction rules. Yarowsky (1994) 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 useful semantic regularities to be exploited for IE.

7.5 Converting BWI Match Scores into Probabilities

While improving the accuracy of IE methods is an important goal, progress will be limited in its usefulness by the output representation used. As mentioned in Section 6.2, BWI (like most other IE methods) does not attach probabilities to the matches it returns. Probability is the *lingua franca* for combining information processing systems, because probabilities have both absolute and relative

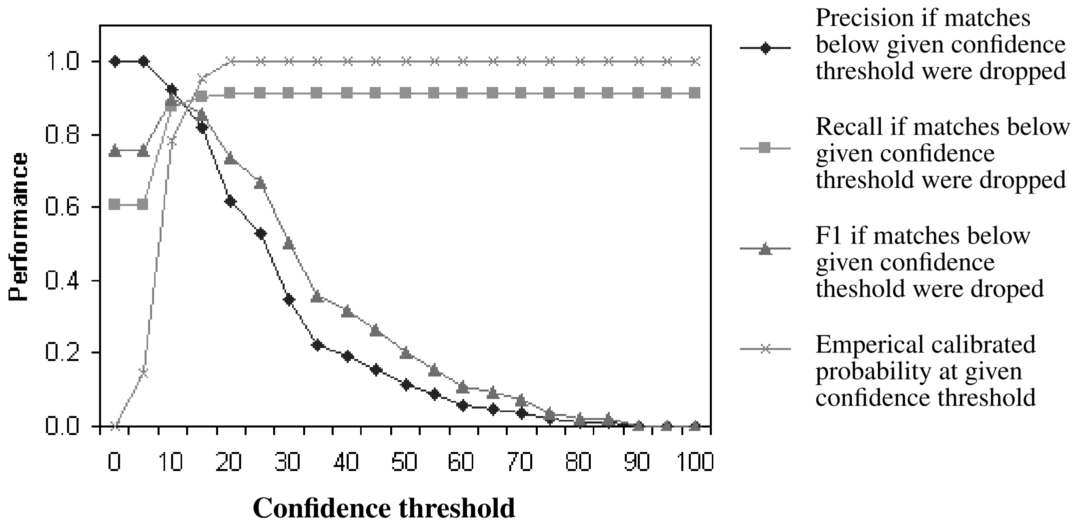

Figure 6: Precision, recall, and F1 for BWI on the *AbstractText* task, versus confidence threshold below which to ignore matches. The empirical probability of match correctness within each confidence interval is also shown.

meaning, and because there are powerful mathematical frameworks for dealing with them, such as Bayesian evidence combination and decision theory. For example, question-answering systems commonly use probabilities (Ponte and Croft, 1998).

Thus, it is worth asking the question, can BWI confidence scores be transformed into probabilities? This question really comes in two pieces. First, are BWI 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 the correlation between confidence scores and the chance of correctness of a match, and thus automatically calibrate its scores into probabilities?

The first question, whether BWI scores are consistent, can be answered empirically, by training and testing BWI on different document collections. Ideally, most incorrect predictions have low confidence compared to correct predictions. Such a distribution is desirable for two reasons. First, it resembles a true probability distribution, where a higher score is correlated with a higher chance of being correct. Second, it allows users to set a confidence threshold above which to accept predictions, and below which to examine predictions further.

Our results suggest that BWI scores are fairly consistent and amenable to threshold setting for tasks with high regularity, but on hard tasks it is difficult to separate correct and incorrect predictions based only on match confidences. For the *AbstractText* task, confidence scores range from 0 to 100, but all incorrect predictions have confidence scores below 20 (Figure 6). This means that if BWI ignores predictions below a confidence threshold of 20, it obtains perfect precision. However, this threshold results in only 59% recall, compared with 95% recall with threshold 0. A confidence threshold of 10 maximizes F1, because most incorrect answers can be pruned without eliminating correct guesses, as can be seen in Figure 6. The confidence scores are behaving roughly as probabilities should, because as confidence increases, so does the fraction of correct guesses.

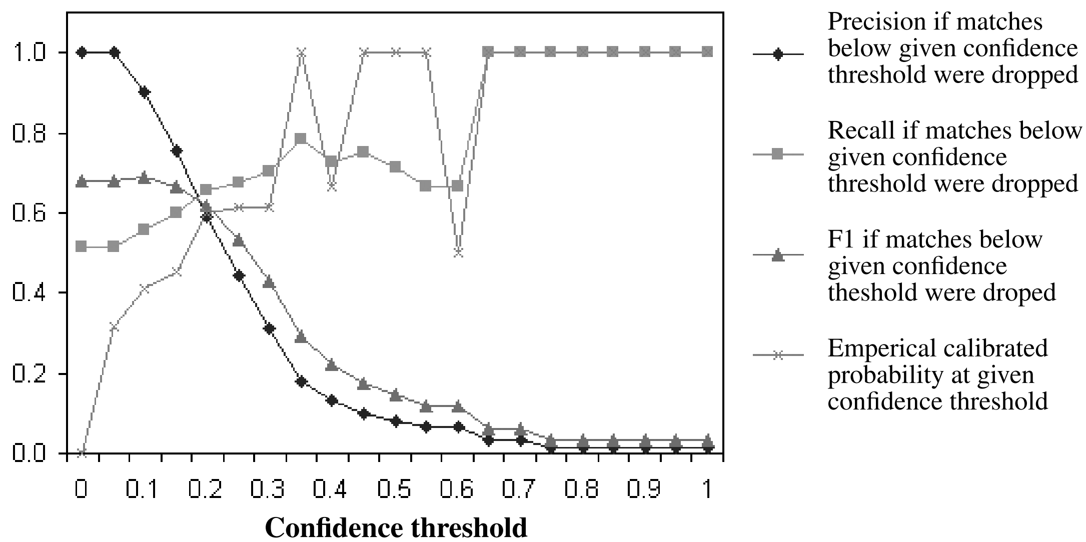

Figure 7: Precision, recall, and F1 for BWI on the *YPD-protein* task, versus confidence threshold below which to ignore matches. The empirical probability of match correctness within each confidence interval is also shown.

Note that the precision for this task is lower than reported in Table 1 because here we consider all BWI matches, whereas normally BWI eliminates overlapping matches, keeping only the match with higher confidence.

Unfortunately, the correlation between confidence and probability of correctness is weaker for more difficult tasks. On the protein task, the most difficult as measured both by performance and by SWI ratio, Figure 7 shows that there is no clean way to separate the correct and incorrect guesses. The highest confidence negative match has score 0.65, but above this threshold, BWI only achieves 0.8% recall, because almost all correct matches have confidence score below 0.35. The highest F1 in this case comes from a confidence threshold of 0, with precision 52% and recall 24%. (These numbers are taken from an individual train/test fold and thus differ slightly from the averages presented in Table 1.) There is no way to improve performance with a non-zero confidence threshold, because there is no smooth transition to a higher density of correct predictions as confidence scores increase.

There is a direct way of calibrating scores into probabilities–divide BWI predictions on a validation set of documents into bins by confidence score, and estimate the probability for each bin as the fraction of correct predictions in the bin. Essentially if for a given range of confidence scores, say 75% of predictions are correct, then a prediction on a test document with similar confidence is estimated to have a 75% chance of being correct. Even for difficult tasks for which confidence scores are not consistent, i.e. as scores increase there is no smooth increase in probability, the calibrated probabilities are still meaningful, because they make explicit the uncertainty in predictions.

		BWI			Fixed-BWI			Root-SWI			Greedy-SWI		
Data set	SWI-ratio	Prec.	Recall	F1	Prec.	Recall	F1	Prec.	Recall	F1	Prec.	Recall	F1
LATimes-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
Zagat-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 Raw histogram	NA	0.860	0.616	0.716	NA	NA	NA	NA	NA	NA	NA	NA	NA
AbstractText No binning	0.149	0.990	0.967	0.978	0.966	0.495	0.654	0.846	0.585	0.685	0.847	0.317	0.447
AbstractText 10-bins	NA	0.976	0.965	0.971	NA	NA	NA	NA	NA	NA	NA	NA	NA
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 tasks.

8. Concluding Remarks

Current information extraction methods are able to discover 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 their regularity is at the surface level-in the use and ordering of words themselves. However, current IE methods perform considerably worse when dealing with natural text documents, because the regularities in these are less apparent. Nevertheless, there *are* still important regularities in natural text documents, at the grammatical and semantic levels. We have shown that making explicit even limited grammatical information results in considerably higher precision and recall for information extraction from natural text documents.

All IE methods perform differently on document collections of different regularity, but there has been no quantitative analysis in previous papers of the relationship between document regularity and IE accuracy. We propose the SWI ratio as a quantitative measure of document regularity, and we use it to investigate the relationship between regularity and accuracy in greater detail than previously possible. Since the SWI ratio objectively measures the relative regularity of a document collection, it is suitable for comparison of IE tasks on document collections of varying size and content.

While all the IE algorithms we study perform worse on less regular document collections, they still exhibit consistent differences. Many current rule-based IE methods, including the SWI method proposed in this paper, employ some form of set covering to combine multiple extraction rules. These methods are relatively simple to implement, but they are all fundamentally limited in that they remove positive examples as soon as they are covered once. Boosting overcomes this limitation by reducing the weight of 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 cov-

	cc	addr	date	abst.	speak.	loc.	stime	etime	id	comp.	title	prot.	loc.	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.

ered, and consistently outperforms set covering methods. Reweighting also helps by focusing BWI on learning specific rules for the exceptional cases missed by general rules, resulting in higher precision. The combination of learning accurate rules, and continuing training to broaden coverage, is the source of the success of BWI.

While this paper focuses on variants of BWI, many of our observations hold for information extraction as a whole. For example, while hidden Markov models employ a different representation, they tend, like BWI, to learn regularities that are local and flat, concerning word usage, punctuation usage, and other surface patterns. Thus the sections above about sources of information that are currently ignored are relevant to HMMs and rule-based IE methods. There are many opportunities for improving the usefulness of current information extraction methods. We hope this paper is a contribution in this direction.

Acknowledgments

The authors would like to thank Dayne Freitag for his assistance 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. The anonymous reviewers provided very detailed and valuable comments that have helped us improve the paper considerably. This work was conducted with support from the California Institute for Telecommunications and Information Technology, Cal-(IT)^2.

References

D. Appelt, J. Hobbs, J. Bear, D. Israel, M. Kameyama, A. Kehler, D. Martin, K. Myers, and M. Tyson. SRI international FASTUS system: MUC-6 test results and analysis. In *Proceedings of the 6th Message Understanding Conference*, pages 237–248, 1995.

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

D. Bikel, S. Miller, R. Schwartz, and R. Weischedel. NYMBLE: a high-performance learning name-finder. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 194–201, 1997.

M. Califf. *Relational Learning Techniques for Natural Language Information Extraction*. PhD thesis, Department of Computer Science, Univeristy of Texas at Austin, 1998.

M. Califf and R. Mooney. Relational learning of pattern-match rules for information extraction. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 328–334, 1999.

C. Cardie and R. Mooney. Tutorial on symbolic machine learning for natural language processing. In *ACL-98*, 1998. URL `http://www.cs.cornell.edu/Info/People/cardie/tutorial/tutorial.html`.

E. Charniak. Immediate-head parsing for language models. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, 2001.

F. Ciravegna. $(LP)^2$, an adaptive algorithm for information extraction from web-related texts. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, 2001.

P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning 3*, pages 261–283, 1989.

W.W. Cohen and Y. Singer. A simple, fast, and effective rule learner. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, 1999.

T. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. In *Neural Computation*, pages 1895–1923, 1998.

T. Eliassi-Rad and J. Shavlik. A theory-refinement approach to information extraction. In *Proceedings of the 18th International Conference on Machine Learning*, 2001.

C. Elkan. The foundations of cost-sensitive learning. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 973–978, 2001.

D. Freitag and N. Kushmerick. Boosted wrapper induction. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 577–583, 2000.

R. Grishman. Information extraction: Techniques and challenges. *Springer-Verlag, Lecture Notes in Artificial Intelligence, Rome*, 1997.

N. Kushmerick. Wrapper induction: Efficiency and expressiveness. In *Artificial Intelligence*, volume 118, pages 15–68, 2000.

L. Liu, C. Pu, and W. Ilan. XWRAP: An XML-enabled wrapper construction system for web information sources. In *Proceedings of the International Conference on Data Engineering*, 2000.

D. Margineantu and T. Dietterich. Pruning adaptive boosting. In *Machine Learning: Proceedings of the Fourteenth International Conference*, pages 211–218, 1997.

S. Michalski. Pattern recognition as rule-guided inductive inference. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 2, pages 349–361, 1980.

G. Miller. Wordnet: A lexical database for english. In *Communications of the ACM*, volume 38, pages 39–41, 1995.

I. Muslea, S. Minton, and C. Knoblock. A hierarchical approach to wrapper induction. In *Proceedings of the Third International Conference on Autonomous Agents*, pages 190–197, 1999.

National Library of Medicine, 2001. Medline database. URL `http://www.ncbi.nlm.gov/Pubmed/`.

J. Ponte and W. Croft. A language modeling approach to information retrieval. In *Research and Development in Information Retrieval*, pages 275–281, 1998.

J. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

S. Ray and M. Craven. Representing sentence structure in hidden markov models for information extraction. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, 2001.

L. Rendell and H. Cho. Empirical learning as a function of concept character. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1044–1049, 1990.

E. Riloff. Automatically generating extraction patterns from untagged text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1044–1049, 1996.

RISE. A repository of online information sources used in information extraction tasks, 1998. URL `http://www.isi.edu/info-agents/RISE/`.

R. Schapire. A brief introduction to boosting. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, 1999.

H. Shinnou. Detection of errors in training data by using a decision list and adaboost. In *IJCAI-2001 Workshop on Text Learning: Beyond Supervision*, pages 61–65, 2001.

S. Soderland, D. Fisher, J. Aseltine, and W. Lehnert. CRYSTAL: Inducing a conceptual dictionary. In *IJCAI'95 - Proc. of the 14th Intl. Joint Conf. on Artificial Intelligence*, volume 2, pages 1314–1319, 1995.

V. Vapnik, E. Levin, and Y. LeCun. Measuring the VC dimension of a learning machine. *Neural Computation*, 6:851–876, 1994.

D. Yarowsky. Decision lists for lexical ambiguity resolution: Application to accent restoration in spanish and french. In *Proceedings of the ACL '94*, pages 77–95, 1994.

W-T. Yih. Template-based information extraction from tree-structured HTML documents. Master's thesis, National Taiwan University, 1997.