Learning to Construct Knowledge Bases from the World Wide Web ⋆

Mark Craven [a] Dan DiPasquo [a] Dayne Freitag [b]
Andrew McCallum [a,b] Tom Mitchell [a] Kamal Nigam [a]
Seán Slattery [a]

[a] *School of Computer Science, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA, 15213-3891, USA*

[b] *Just Research, 4616 Henry Street, Pittsburgh, PA, 15213, USA*

Abstract

The World Wide Web is a vast source of information accessible to computers, but understandable only to humans. The goal of the research described here is to automatically create a *computer understandable* knowledge base whose content mirrors that of the World Wide Web. Such a knowledge base would enable much more effective retrieval of Web information, and promote new uses of the Web to support knowledge-based inference and problem solving. Our approach is to develop a trainable information extraction system that takes two inputs. The first is an ontology that defines the classes (e.g., company, person, employee, product) and relations (e.g., employed_by, produced_by) of interest when creating the knowledge base. The second is a set of training data consisting of labeled regions of hypertext that represent instances of these classes and relations. Given these inputs, the system learns to extract information from other pages and hyperlinks on the Web. This article describes our general approach, several machine learning algorithms for this task, and promising initial results with a prototype system that has created a knowledge base describing university people, courses, and research projects.

⋆ This research has been supported in part by the DARPA HPKB program under research contract F30602-97-1-0215. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of DARPA or the U.S. government.

1 The Opportunity

The rise of the World Wide Web has made it possible for your workstation to retrieve over 350,000,000 Web pages for your personal perusal. The Web has already become one of the largest and most diverse sources of information on the planet, and many expect it to grow into the world's primary knowledge resource over the next decade.

The research described here is motivated by a simple observation: although your workstation can currently *retrieve* over 350,000,000 Web pages, it currently *understands* none of these Web pages. The goal of our WEB→KB research project is to automatically create a computer-understandable knowledge base whose content mirrors that of the World Wide Web. Such a "World Wide Knowledge Base" would consist of computer understandable assertions in symbolic, probabilistic form (e.g., employed_by(mark_craven, carnegie_mellon_univ), probability=.99). We expect such a world wide knowledge base would have many uses. At a minimum, it would allow much more effective information retrieval by supporting queries such as "find all universities within 20 miles of Pittsburgh that offer evening courses on Java programming." Going a step further, it would enable new uses of the Web to support knowledge-based inference and problem solving. For example, it would provide the knowledge base needed by a software travel agent that might handle requests such as "make me hotel and flight arrangements for the upcoming ACM conference." Notice that information about the ACM conference, nearby hotels, and flights is already available in human-readable form, spread across multiple text pages on the Web. A knowledge base that makes this information computer-understandable would support a variety of intelligent knowledge-based agents.

How might we construct and maintain such a world wide knowledge base? The thesis explored in this article is that one can develop such a knowledge base by (1) using machine learning to create information extraction methods for each of the desired types of knowledge, then (2) applying these learned information extraction methods to extract symbolic, probabilistic statements directly from Web hypertext. Each assertion in the knowledge base can therefore carry with it a justification, in terms of the Web sources and information extraction method, that provide its evidential support. As the Web evolves over time, the knowledge base can be automatically updated to reflect its changing content.

This article explores the above thesis by proposing and evaluating several learning algorithms relevant to this information extraction task, and by presenting the prototype WEB→KB system which has successfully built a knowledge base containing several thousand assertions about computer science departments using these learned information extractors.

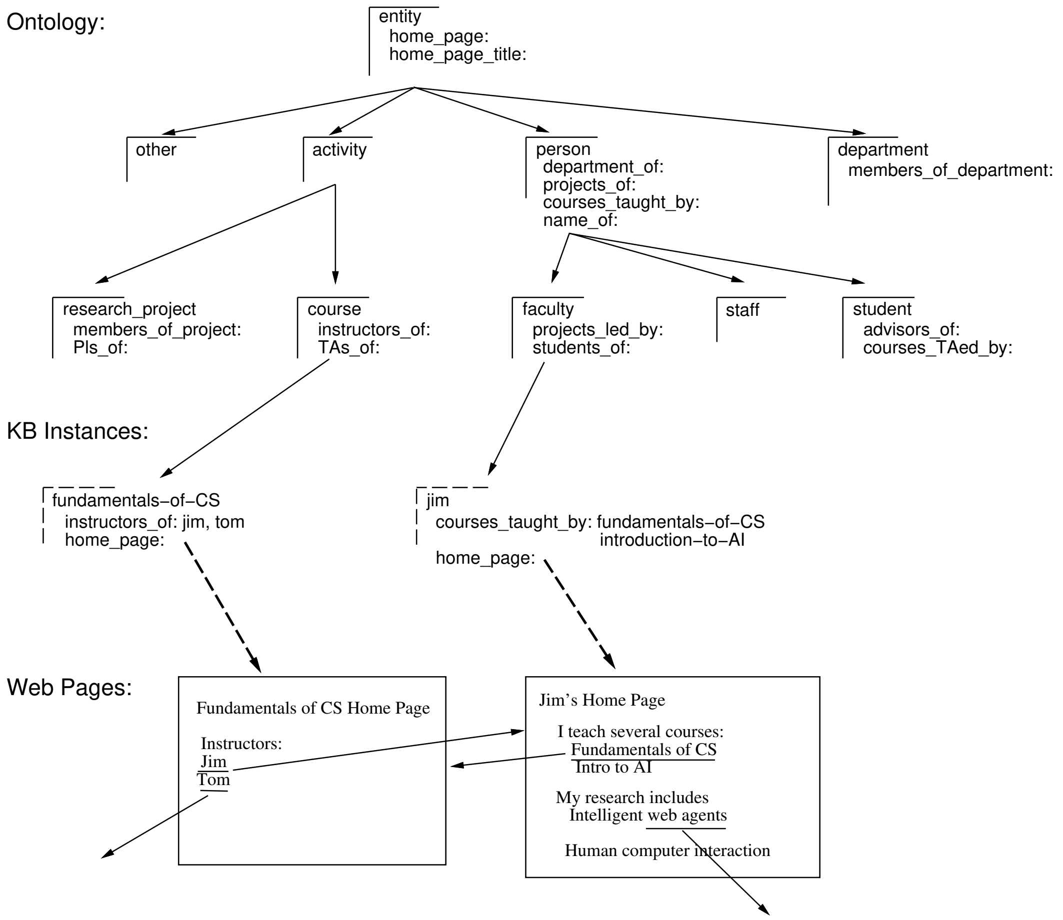

Fig. 1. An overview of the WEB→KB system. The top part of the figure shows an ontology that defines the classes and relations of interest. Each partial box represents a class, and the arrows indicate specialization relationships. The other defined relations for each class are listed inside of its corresponding box. The bottom part shows two Web pages identified as training examples of the classes course and faculty. Together,these two pages also constitute a training example for the relations instructors_of and courses_taught_by. Given the ontology and a set of training data, WEB→KB learns to interpret additional Web pages and hyperlinks to add new instances to the knowledge base, such as those shown in the middle of the figure. These instances are represented by dashed partial boxes. The dashed lines show the relationships between the instances and their Web sources.

We begin by briefly surveying the capabilities of the WEB→KB system in the next section. The subsequent section considers in detail the representational assumptions underlying our approach. The remaining sections present our experimental testbed, several learning algorithms and experimental results for the various information extraction tasks, related work, and conclusions.

2 Overview of the WEB→KB System

The WEB→KB system is first trained to extract information of the desired types, and is then allowed to browse new Web sites in order to automatically populate a knowledge base with new assertions. When training this system, the user must provide two inputs:

(1) A specification of the classes and relations of interest. This is the *ontology* that defines the vocabulary for the target knowledge base. An example of such an ontology is provided in the top half of Figure 1. This particular ontology defines a hierarchy of classes including person, student, research_project, course, etc. It also defines relations between these classes such as advisors_of (which relates an instance of a student to the instances of faculty who are the advisors of the given student). This ontology, and optionally instances of some of the classes and relations, constitutes the initial version of the knowledge base.

(2) Training examples that describe instances of the ontology classes and relations. For example, the two Web pages shown at the bottom of Figure 1 represent instances of course and faculty classes. Furthermore, this pair of pages represents an instance of the relation courses_taught_by (i.e., the courses_taught_by jim includes fundamentals-of-CS).

Given such an ontology and a set of training examples, the WEB→KB system learns general procedures for extracting new instances of these classes and relations from the Web. In the current prototype, this is accomplished by learning to classify arbitrary Web pages and hyperlink paths according to the classes and relations defined by the ontology. When exploring the Web, the WEB→KB system starts from a given input URL and explores pages using a breadth-first search to follow links. Each explored page is examined, using learned class descriptions, to see if it represents a member of one of the ontology classes. If a page is determined to be a class member, an entity representing that page is placed into the knowledge base, and the ontology relations for that page are instantiated based on learned rules and the local structure of the Web around the page. If a page is not a class member, the search is truncated, and links from this page are not followed.

In one such crawling experiment the system was given a training set of approximately 8,000 Web pages and 1,400 Web-page pairs taken from the computer science department Web sites at four universities (Cornell, University of Texas at Austin, University of Washington, and University of Wisconsin). These training examples were hand labeled according to the ontology shown in Figure 1. The system was then allowed to explore the Web site of a fifth computer science department (at Carnegie Mellon University), and to add new knowledge base entries based on information extracted from this new Web site.

```
david-garlan
  generalizations: faculty
  name: "David Garlan"
  courses_taught_by: cmu-cs-15-675-architectures-of-software-systems
  projects_of: architectural-mismatch
                   cmu-cs-composable-software-systems-home-page
                   able-project
  department_of: school-of-computer-science
  home_page: "http://www.cs.cmu.edu/afs/cs/user/garlan/www/home.html"

able-project
  generalizations: research_project
  members_of_project: bridget-spitznagel
                                 david-garlan
                                 ralph-melton
                                 heather-richter
  home_page: "http://www.cs.cmu.edu/afs/cs/project/able/www/index.html"
```

Fig. 2. Two of the entities automatically extracted from the CMU computer science department Web site after training on four other university computer science sites. These entities were added as new instances of faculty and project to the knowledge base shown in Figure 1. The identifiers used to reference entities (e.g., david-garlan and able-project) are automatically generated from the titles of the pages from which the entities are extracted.

	student	faculty	person	research_project	course	department	**Overall**
Extracted	180	66	246	99	28	1	374
Correct	130	28	194	72	25	1	256
Accuracy	72%	42%	79%	73%	89%	100%	68%

Table 1
Class instance recogni tion accuracy when exploring CMU computer science department Web site, after training on computer science departments at four other universities.

Two new instances added by the system to its knowledge base, as a result of browsing this new university Web site, are shown in Figure 2. The top instance describes a new faculty added to the knowledge base, as a result of examining a Web page that the system classified into this category. As a result, the system created the new faculty instance in the knowledge base, and extracted several relations involving this instance. For example, it determined (correctly) the name of this faculty member, a course taught by the faculty member, and three instances of the projects_of relation for this faculty member: architectural-mismatch, cmu-cs-composable-software-systems-home-page, and able-project. These three projects are themselves instances of the research_project class, extracted from other Web pages. The description of one of these, the able-project, is shown at the bottom of the figure. The identifiers used to reference these instances are automatically generated from the titles of the pages from which the instances are extracted.

How accurate is the system in extracting such information? In this experiment, the system visited 2722 Web pages at the new Carnegie Mellon site, and as

	instructors_of	members_of_project	department_of	**Overall**
Extracted	23	125	213	361
Correct	18	92	181	291
Accuracy	78%	74%	85%	81%

Table 2
Relation instance recognition accuracy when exploring CMU computer science department Web site, after training on computer science departments at four other universities.

a result added 374 new class instances to its knowledge base. The fraction of correctly extracted instances is summarized in Table 1. For example, this table indicates that the system created 28 new knowledge base instances of the class course. Of these 28 new instances, 25 in fact represented courses and the other 3 did not. Its accuracy [1] in extracting relation instances is summarized in a similar fashion in Table 2. Note that since we don't have a labelling for all the pages and relations at Carnegie Mellon, we have no way of calculating coverage results for these tasks.

Figure 3 shows the information displayed by the system as it browses. Note this display shows the Web page that is currently being visited (top right), and the information extracted by the system from this Web page (middle left). The interface also contains a control panel that allows the user to interact with the system (top left), and to browse the growing knowledge base (bottom right).

3 Problem Formulation

As summarized in the previous section, the WEB→KB system provides experimental support for our thesis that a system can be trained to automatically populate a knowledge base by browsing Web hypertext. Later sections describe in detail the learning algorithms used by this WEB→KB system. In this section, we consider the precise problem formulation and the representational assumptions that underlie our current approach.

To summarize, we are interested in the following general problem:

Given:

· a knowledge base consisting of an ontology defining the classes (e.g., person) and relations (e.g., instructor_of) of interest, and optionally, instances

[1] Throughout the article, when we refer to *accuracy* on a per-class basis, it is equivalent to the *precision* measure commonly used in the information-retrieval community. Similarly, when we refer to *coverage* on a per-class basis, it is equivalent to the *recall* measure.

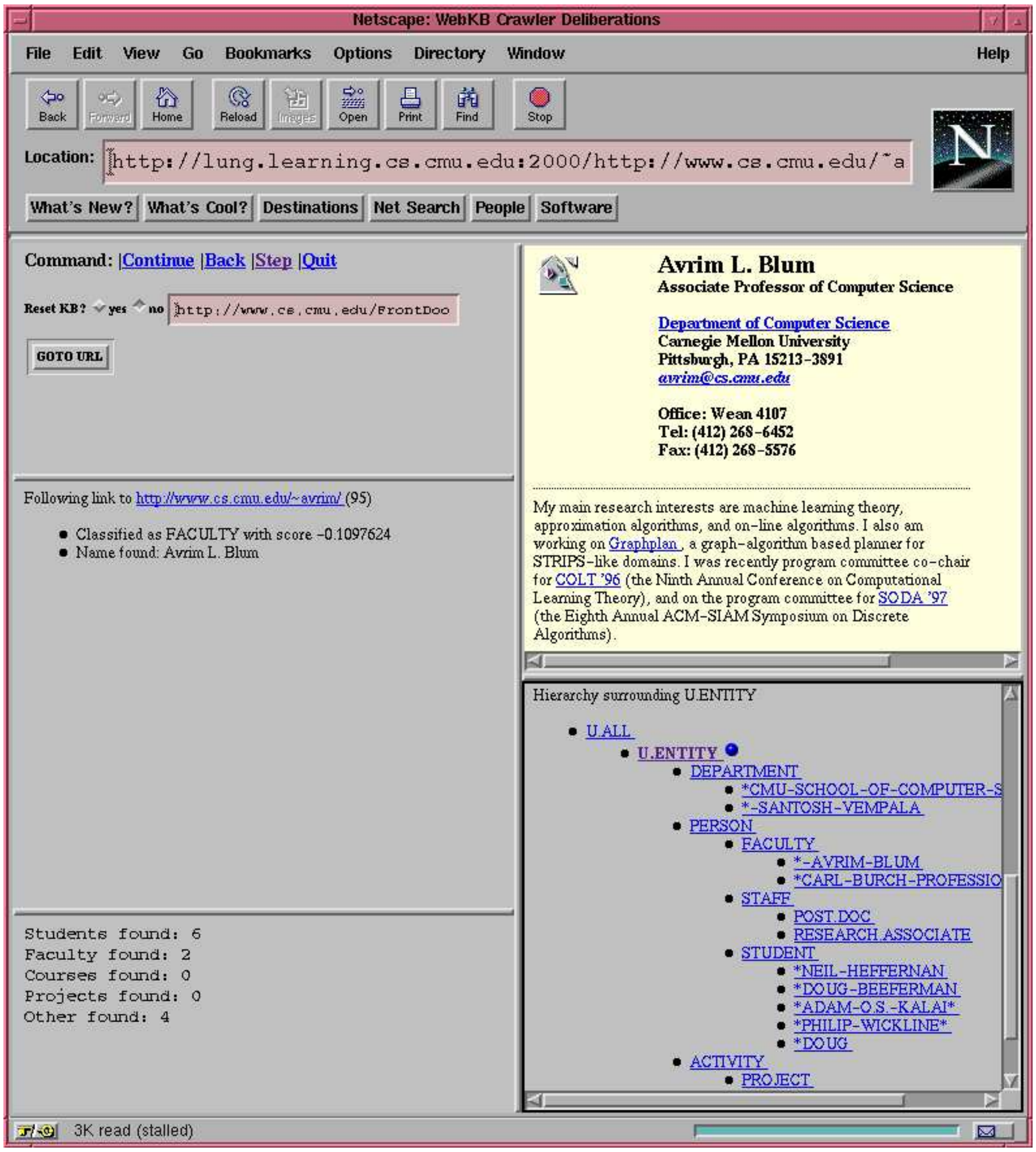

Fig. 3. The Web interface to the WEB→KB system. The upper left pane serves as a control panel. The middle left pane describes the current activity of the WEB→KB system, and the lower left pane summarizes the session. The upper right pane shows the page currently being processed, and the lower right pane provides a mechanism for browsing the extracted knowledge base.

of some of these classes and relations,

· training examples from the Web that describe instances of these classes and relations.

Determine:

· general procedures capable of extracting additional instances of these classes and relations by browsing the rest of the Web.

Note that we do not necessarily extract new instances for *all* of the classes and relations in the ontology. For example, our ontology may have a class country and the initial knowledge base may include instances for all of the countries in the world. In this case, since we already know all instances, we do not need to learn procedures to recognize new ones.

To pursue the problem of learning to extract instances from the Web, we must make some assumptions about the types of knowledge to be extracted from the Web, and the way in which this knowledge is represented in hypertext on the Web. These assumption are:

- *Assumptions about how class instances are described on the Web.* We assume that each instance of an ontology class is represented by one or more *segments* of hypertext on the Web. By segment of hypertext, we mean either a single Web page, or a contiguous string of text within a Web page, or a rooted graph of several Web pages linked by the directed edges of hyperlinks. For example, an instance of a person might be described by a single page (the person's home page), or by a reference to the person in a string of text within an arbitrary Web page, or by a collection of interconnected Web pages that jointly describe the person.
- *Assumptions about how relation instances are described on the Web.* Consider an arbitrary instance R(A,B) of a relation R. We assume that each instance of a relation is represented on the Web in one of three ways. First, the instance R(A,B) may be represented by an undirected path of hyperlinks and pages that connects the segment representing A to the segment representing B. For example, the bottom of Figure 1 shows two hyperlinks that connect the segment representing jim to the segment representing fundamentals-of-CS. These hyperlinks represent the relation instructor_of(fundamentals-of-CS, jim). Second, the instance R(A,B) may alternatively be represented by a segment of text representing A that *contains* the segment that represents B. For example, the relation instance courses_taught_by(jim, introduction-to-AI) is represented in Figure 1 by the fact that Jim's home page contains the phrase "Intro to AI" in a particular context. Finally, the instance R(A,B) may be represented by the fact that the hypertext segment for A satisfies some learned model for relatedness to B. For example, we might extract the instance research_area_of(jim, artificial-intelligence) by classifying Jim's page using a statistical model of the words typically found in pages describing AI research.

Appendix A provides a more formal presentation of these assumptions. In addition to these assumptions about the mapping between Web hypertext and the ontology, we make several simplifying assumptions in our initial research reported in this article. We plan to relax the following assumptions in the future as our research progresses.

- We assume in this article that each class instance is represented by a single

Web page (e.g., a person is represented by their home page). If an instance happens to be described by multiple pages (e.g., if a person is described by their home page plus a collection of neighboring pages describing their publications, hobbies, etc.), our current system is trained to classify only the primary home page as the description of the person, and to ignore the neighboring affiliated pages. Alternatively, if an instance happens to be described by a text fragment, our system does not currently create a knowledge base instance for this. It does, however, extract certain relation values from such text fragments (e.g., the name of the person, as illustrated in Figure 1).

- We assume that each class instance is represented by a single segment of hypertext. In other words, if the system encounters two unlinked Web pages that represent instances of the same class, it creates two distinct instances of this class in its knowledge base. While this assumption will often be satisfied (e.g., two distinct personal home pages typically represent two distinct people), there are clearly exceptions (e.g., there are many different Web pages describing Elvis). Overcoming this "multiple Elvis problem" will require methods that hypothesize equivalences between independently discovered instances.
- We assume that all relations are two-place relations; that is, each relation has only two arguments. We believe that it will be fairly easy to relax this assumption.

Given this problem definition and our current set of assumptions, we view the following as the three primary learning tasks that are involved in extracting knowledge-base instances from the Web:

(1) Recognizing class instances by classifying bodies of hypertext. Section 5 looks at this problem, using both statistical and relational learning techniques. It also examines how to relax our assumption about class instances being represented by single Web pages.
(2) Recognizing relation instances by classifying chains of hyperlinks. Section 6 investigates a relational learning solution to this problem.
(3) Recognizing class and relation instances by extracting small fields of text from Web pages. Section 7 looks at this task and also uses a relational learning approach.

4 Experimental Testbed

All experiments reported in this article are based on the ontology for computer science departments shown in Figure 1. This ontology includes the classes department, faculty, staff, student, research_project, and course. Our Web page classification experiments also use the class other as the label for Web pages that fall into none of these ontology classes. Each ontology class has an associated set of slots, or relations, that exist among instances of this class and other class instances in the ontology. For example, the course class has a slot called instructors_of that relates courses to people.

We assembled two data sets [2] for the experiments reported here. The first is a set of pages and hyperlinks drawn from four CS departments: University of Texas at Austin, Cornell University, University of Washington, and University of Wisconsin. The second is a set of pages from numerous other computer science departments. The four-department set includes 4,127 pages and 10,945 hyperlinks interconnecting them. The second set includes 4,120 additional pages. The pages for most of the classes in our data set were collected using "index" pages for our classes of interest (e.g., a page that has hyperlinks to all of the students in a department), so labeling this data was straightforward. After gathering this initial set of pages, we then collected every page that was both (i) pointed to by a hyperlink in the initial set, and (ii) from the same university as the page pointing to it. Most of the pages gathered in the second step were labeled as other.

In addition to labeling pages, we also labeled relation instances. Each of these relation instances consists of a pair of pages corresponding to the class instances involved in the relation. For example, an instance of the instructors_of relation consists of a course home page and a person home page. Our data set of relation instances comprises 251 instructors_of instances, 392 members_of_project instances, and 748 members_of_department instances. These instances are all from the four-department set.

Finally, we also labeled the name of the owner of pages in the person class. This was done automatically by tagging any text fragment in the person's home page that matched the name as it appeared in the hyperlink pointing to the page from the index page. The matching heuristics were conservative, favoring precision over recall. Consequently, we believe that, although some name occurrences were missed, there were no false positives. From 174 person pages, this procedure yielded 525 distinct name occurrences. These instances are all from the four-department set as well.

For all of the subsequent experiments in this article, we use a four-fold cross-validation methodology to evaluate our algorithms. We conduct four runs in which we train classifiers using data from three of the universities in our data set (plus the second set of pages where applicable), and test the classifiers using data from the remaining university. On each iteration we hold out a different university for the test set.

---

[2] These data sets are publicly available at `http://www.cs.cmu.edu/~webkb/`.

5 Learning to Recognize Class Instances

The first task for our system is to identify new instances of ontology classes from the text sources on the Web. In this section we address the case in which class instances are represented by Web pages; for example, a given instance of the `student` class is represented by the student's home page.

In the first part of this section we discuss a statistical bag–of–words approach to classifying Web pages. We use this method along with three different representations of pages. In the second part of this section we discuss learning first-order rules to classify Web pages. An appealing aspect of the second approach is that first-order rules can describe page classes using a rich description of the local graph structure around the page. Finally, we evaluate the effectiveness of combining the predictions made by all four of these classifiers.

5.1 Statistical Text Classification

In this section we consider classifying Web pages using statistical methods. Our approach is similar to a growing body of work in text classification that involves using a so-called *bag of words* or *unigram* representation. However, we apply our method in novel ways that take advantage of the redundancy of hypertext. Specifically, we train three independent classifiers which use different representations for page classification:

- *Full-Text*: the words that occur anywhere in the page,
- *Title/Heading*: the words that occur in the title and HTML headings of the page,
- *Hyperlink*: the words that occur in hyperlinks (i.e., the words in the anchor text) that point to the page.

5.1.1 Approach

Our approach involves building a probabilistic model of each class using labeled training data, and then classifying newly seen pages by selecting the class that is most probable given the evidence of words describing the new page.

The method that we use for classifying Web pages is naive Bayes, with minor modifications based on Kullback-Leibler Divergence. Given a document d to classify, we calculate a score for each class c as follows:

$$Score_c(d) = \frac{\log \Pr(c)}{n} + \sum_{i=1}^{T} \Pr(w_i|d)\ \log\left(\frac{\Pr(w_i|c)}{\Pr(w_i|d)}\right) \qquad (1)$$

where n is the number of words in d, T is the size of the vocabulary, and w_i is the ith word in the vocabulary. $Pr(w_i|c)$ thus represents the probability that a randomly drawn word from a randomly drawn document in class c will be the word w_i. $Pr(w_i|d)$ represents the proportion of words in document d that are word w_i. The class predicted by the method for a given document is simply the class with the greatest score. This method makes exactly the same classifications as naive Bayes, but produces classification scores that are less extreme. Below we explain naive Bayes; in Appendix B we detail our modifications to it.

Naive Bayes

The probabilistic models we use ignore the sequence in which the words occur. Such models are often called *unigram* or *bag-of-words* models because they are based on statistics about single words in isolation.

Since the unigram model naively assumes that the presence of each word in a document is conditionally independent of all other words in the the document given its class, this approach, when used with Bayes Rule is often called *naive Bayes*. The conditional independence assumption is clearly violated in real-world data, however, despite these violations, empirically the naive Bayes classifier does a good job of classifying text documents [37,67,27,42]. This observation is in part explained by the fact that classification estimation is only a function of the sign (in binary cases) of the function estimation [17,24]. The word independence assumption causes naive Bayes to give extreme (almost 0 or 1) class probability estimates. However, these estimates can still be poor while classification accuracy remains high.

There are two common approaches to naive Bayes text classification. One, the *multi-variate Bernoulli* model, is a Bayesian Network with no dependencies between words and binary word counts; the document is considered to be the "event" and each feature is a Boolean variable indicating the presence or absence of a particular word in that document. The other approach, the *multinomial* model, is a unigram language model with integer word counts; the words are considered to "events" and the document is comprised of a collection of these events. We use the second approach, since it has been found to outperform the first on several data sets [43].

We formulate naive Bayes for text classification as follows. Given a set of classes $\mathcal{C} = \{c_1, ...c_N\}$ and a document consisting of n words, $(w_1, w_2, ...w_n)$, we classify the document as a member of the class, $c^\star$, that is most probable, given the words in the document:

$$c^\star = \operatorname{argmax}_c \Pr(c|w_1, ..., w_n). \tag{2}$$

We transform $\Pr(c|w_1, ..., w_n)$ into a computable expression by applying Bayes Rule (Eq. 3); rewriting the expression using the product rule and dropping the denominator, since this term is a constant across all classes, (Eq. 4); and assuming that words are independent of each other (Eq. 5).

$$\begin{aligned}
\Pr(c|w_1, ..., w_n) &= \frac{\Pr(c)\Pr(w_1, ..., w_n|c)}{\Pr(w_1, ..., w_n)} && (3)\\
&\propto \Pr(c)\prod_{i=1}^{n}\Pr(w_i|c, w_1, ..., w_{i-1}) && (4)\\
&\simeq \Pr(c)\prod_{i=1}^{n}\Pr(w_i|c) && (5)
\end{aligned}$$

The modifications that transform this traditional formulation of naive Bayes into the form we use (shown in Equation 1) are described in Appendix B.

Estimating Word Probabilities

A key step in implementing naive Bayes is estimating the word probabilities, $\Pr(w_i|c)$. To make our probability estimates more robust with respect to infrequently encountered words, we use a *smoothing* method to modify the probabilities that would have been obtained by simple event counting. One important effect of smoothing is that it avoids assigning probability values of zero to words that do not occur in the training data for a particular class. Since naive Bayes involves taking a product of word probabilities, a single zero for a class would prevent that class from being the maximum even if there are many other words that strongly indicate that class. Rather than smoothing with the common Laplace Estimates (*i.e.*, adding one to all the word counts for a class), we use Witten-Bell smoothing [66], which we have found to perform better in some cases, particularly when the amount of training data in each class differs dramatically. Witten-Bell differs from Laplace smoothing in that it strength of the prior depends on the relationship between the number of unique words and the total number of word occurrences in the training data for the class; if most of the word occurences are unique words, the prior is stronger, if words are often repeated, the prior is weaker. More precisely

Witten-Bell sets $\Pr(w_i|C)$ as follows:

$$\Pr(w_i|c) = \begin{cases} \frac{N(w_i,c)}{T_c+\sum_j N(w_j,c)} & \text{if } N(w_i,c) \neq 0 \\ \frac{T}{T_c+\sum_j N(w_j,c)} \frac{1}{T-T_c} & \text{if } N(w_i,c) = 0 \end{cases} \quad (6)$$

where $N(w_i, c)$ is the count of the number of times word w_i occurs in the training data for class c, T_c is the total number of unique words in class c, and T is the total number of unique words across all classes. Note that if we set T and T_c to zero, (and define $0/0 = 0$), we obtain the standard maximum likelihood estimates for $\Pr(w_i|c)$.

Feature Selection

Another important implementation issue is deciding upon the vocabulary size to be used for the problem domain. We have found empirically that we get slightly more accurate classifications when using a restricted vocabulary size. Thus we limit our vocabulary to 2000 words in all of our experiments. The vocabulary is selected by ranking words according to their average mutual information with respect to the class labels [14]. We write W_i for a random variable indicating whether word w_i is present or absent in a document, and write $v_i \in \{w_i, \neg w_i\}$ for the values it takes on. We write C for a random variable taking values of all the class labels, $c \in \mathcal{C}$. Then, average mutual information is

$$\begin{aligned} I(C;W_i) &= H(C) - H(C|W_i) && (7) \\ &= -\sum_{c\in\mathcal{C}} \Pr(c)\log(\Pr(c)) \\ &\quad - - \sum_{v_i\in\{w_i,\neg w_i\}} \Pr(v_i) \sum_{c\in\mathcal{C}} \Pr(c|v_i)\log(\Pr(c|v_i)) \\ &= \sum_{v_i\in\{w_i,\neg w_i\}} \sum_{c\in\mathcal{C}} \Pr(c,v_i)\log\left(\frac{\Pr(c,v_i)}{\Pr(c)\Pr(v_i)}\right) && (8) \end{aligned}$$

This feature selection method has been found to perform best among several alternatives [67], and has been used in many text classification studies [27,31,32,48,42].

5.1.2 Experimental Evaluation

We evaluate our method using the data sets and cross-validation methodology described in Section 4. On each iteration of the cross-validation run, we train a classifier for each of the page representations described at the beginning of

Predicted	Actual: course	student	faculty	staff	research_project	department	other	Accuracy
course	**202**	17	0	0	1	0	552	26.2
student	0	**421**	14	17	2	0	519	43.3
faculty	5	56	**118**	16	3	0	264	17.9
staff	0	15	1	**4**	0	0	45	6.2
research_project	8	9	10	5	**62**	0	384	13.0
department	10	8	3	1	5	**4**	209	1.7
other	19	32	7	3	12	0	**1064**	93.6
Coverage	82.8	75.4	77.1	8.7	72.9	100.0	35.0	

Table 3
A confusion matrix showing the results of classifying Web pages using the full-text classifier. Results are combined across all of the four-university test runs. The overall coverage and accuracy are also shown.

this section: full-text, title/heading, and hyperlink. Table 3 shows the resulting confusion matrix (summed over the four test sets) for the full-text classifiers. Each column of the matrix represents one class and shows how the instances of this class are classified. Each row represents the instances that are predicted to belong to a given class, and shows the true classes of these instances. This table illustrates several interesting results. First, note that for most classes, coverage is quite high. For example, 83% of the course and 77% of the faculty instances are correctly classified. The notable exception to this trend is the other class; only 35% of the instances belonging to this class are correctly classified. We discuss this result in more detail below. A second interesting result is that many of the remaining mistakes made by the classifiers involve confusing different subclasses of person. For example, although only 9% of the staff instances are correctly assigned to the staff category, 80% of them are correct at the superclass level of person. As this result suggests, not all mistakes are equally harmful; even when we fail to correctly classify an instance into one of the leaf classes in our ontology, we can still make many correct inferences about the instance if we correctly assign it to a more general class.

The low level of classification accuracy for the other class is largely explained by the nature of this class. Recall from Section 4 that the instances of this class were collected by gathering pages that were one hyperlink away from the instances in the other six classes. For this reason, many of the instances of the other class have content, and hence word statistics, very similar to instances in one of the "core" classes. For example, whereas the home page for a course will belong to the course class, "secondary" pages for the course, such as a page describing reading assignments, will belong to the other class. Although the content of many of the pages in the other class might suggest that they properly belong in one of the core classes, our motivation for not including them in these classes is the following. When our system is browsing the Web and adding new instances to the knowledge base, we want to ensure that we do not add multiple instances that correspond to the same real-world object. For example, we should not add two new instances to the knowledge base when we encounter a course home page and its secondary page listing the reading assignments. Because of this requirement, we have framed our page classification task as one of correctly recognizing the "primary" pages for the classes of interest. As Table 3 indicates, this is a very difficult task, but as we will show shortly, by combining several sources of evidence for each page, it is one we can perform with high accuracy.

One way to obtain insight into the learned classifiers is to ask which words contribute most highly to the quantity $Score_c(d)$ for each class. To measure this, we used one of our training sets (consisting of data from three of the four universities) to calculate

$$\Pr(w_i|c)\log\left(\frac{\Pr(w_i|c)}{\Pr(w_i|\neg c)}\right) \tag{9}$$

for each word w_i and class c. Figure 4 shows the ten words for each class that have the greatest value of this weighted log-odds ratio. For space reasons, we do not show the words for the staff class. As the table illustrates, most of the highly weighted words are intuitively prototypical for their class. The exceptions to this generalization are mostly from the other class which represents an extremely diverse set of pages.

Another interesting result illustrated by this table is that many words which are conventionally included in *stop lists*[3] are highly weighted by our models. For example, the words *my*, *me*, and *am* are typical stop-list words but they are among the top ten words for the student class. Although these are common words, they are clearly predictive of the student class since first-person pronouns and verb conjugations do not appear frequently on pages in the other

[3] A *stop list* is a set of words that are commonly removed from documents before they are processed by an information-retrieval or text-classification system. There are standard stop lists which include words generally thought to convey little information about the document topic.

student

my	0.0247
page	0.0109
home	0.0104
am	0.0085
university	0.0061
computer	0.0060
science	0.0059
me	0.0058
at	0.0049
here	0.0046

faculty

DDDD	0.0138
of	0.0113
and	0.0109
professor	0.0088
computer	0.0073
research	0.0060
science	0.0057
university	0.0049
DDD	0.0042
systems	0.0042

course

course	0.0151
DD:DD	0.0130
homework	0.0106
will	0.0088
D	0.0080
assignments	0.0079
class	0.0073
hours	0.0059
assignment	0.0058
due	0.0058

research_project

group	0.0060
project	0.0049
research	0.0049
of	0.0030
laboratory	0.0029
systems	0.0028
and	0.0027
our	0.0026
system	0.0024
projects	0.0020

department

department	0.0179
science	0.0153
computer	0.0111
faculty	0.0070
information	0.0069
undergraduate	0.0058
graduate	0.0047
staff	0.0045
server	0.0042
courses	0.0042

other

D	0.0374
DD	0.0246
the	0.0153
eros	0.0010
hplay*D*	0.0097
u*DD*b	0.0067
to	0.0064
bluto	0.0052
gt	0.0050
that	0.0043

Table 4
The top ten most highly weighted words. For each class, the table shows the ten words that are most highly weighted by one of our learned full-text models. The weights shown represent the weighted log-odds ratio of the words given the class. The symbol D is used to represent an arbitrary digit. For example, the top word shown for the **faculty** class, *DDDD*, represents any four-digit token (such as that occurring in a phone number).

classes. This result suggests that it is advantageous to select a vocabulary in a domain specific way (as we did using mutual information), instead of using a general purpose stop list.

Table 3 shows the results when we assign each page to the class with the highest score. One approach to improving classification accuracy is to limit the

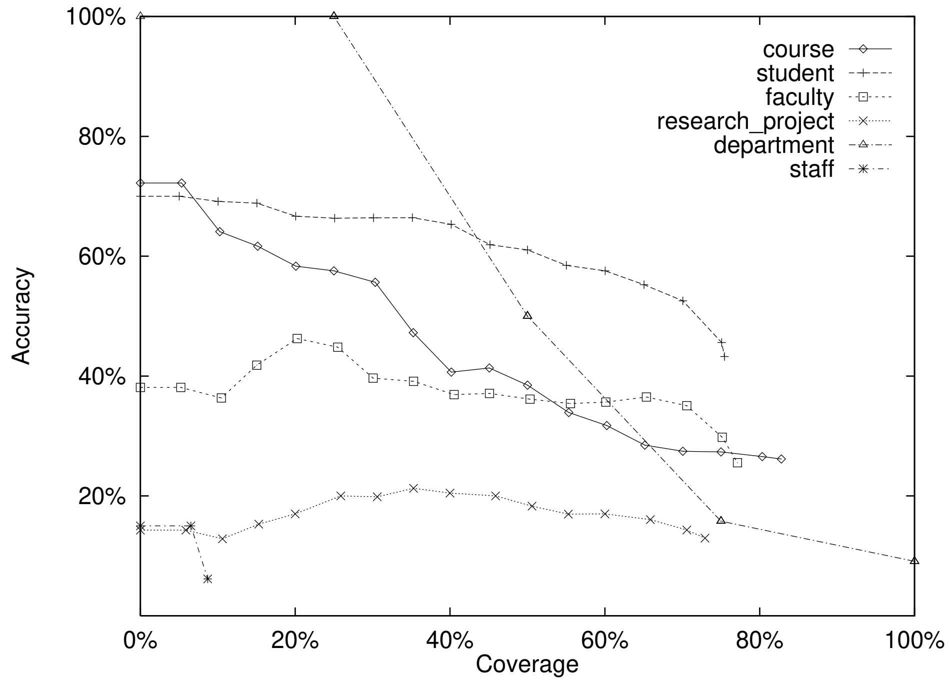

Fig. 4. Accuracy/coverage tradeoff for full-text classifiers. Predictions within each class are ordered according to their confidence. Each curve shows the behavior of the classifier as a threshold on this confidence is varied. The x-axis represents the percentage of pages of a given class that are correctly classified as belonging to the class. The y-axis represents the percentage of pages assigned to a given class that are actually members of that class.

predictions made by the classifiers to just those predictions in which they are most confident. This is easily achieved with our method because the quantity $Score_c(d)$ calculated when classifying a page can be taken as a measure of the confidence in the classification. By setting a minimum threshold on this confidence, we can select a point that sacrifices some coverage in order to obtain increased accuracy. Given our goal of automatically extracting knowledge base information from the Web, it is desirable to begin with a high-accuracy classifier, even if we need to limit coverage to only, say, 10% of the 350,000,000 pages available on the Web.

The effect of trading off coverage for accuracy using our full-text classifiers is shown in Figure 4. The horizontal axis on this plot represents *coverage*: the percentage of pages of a given class that are correctly classified as belonging to the class. The vertical axis represents *accuracy*: the percentage of pages classified into a given class that are actually members of that class. To understand these results, consider, for example, the class `student`. As the results in Table 3 show, when the classifiers predict that a page belongs to the `student` class they are correct 43% of the time. The rightmost point on the `student` curve in the Table 4 corresponds to this point. As we raise the confidence threshold for this class, however, the accuracy of our predictions rises. For example, at a

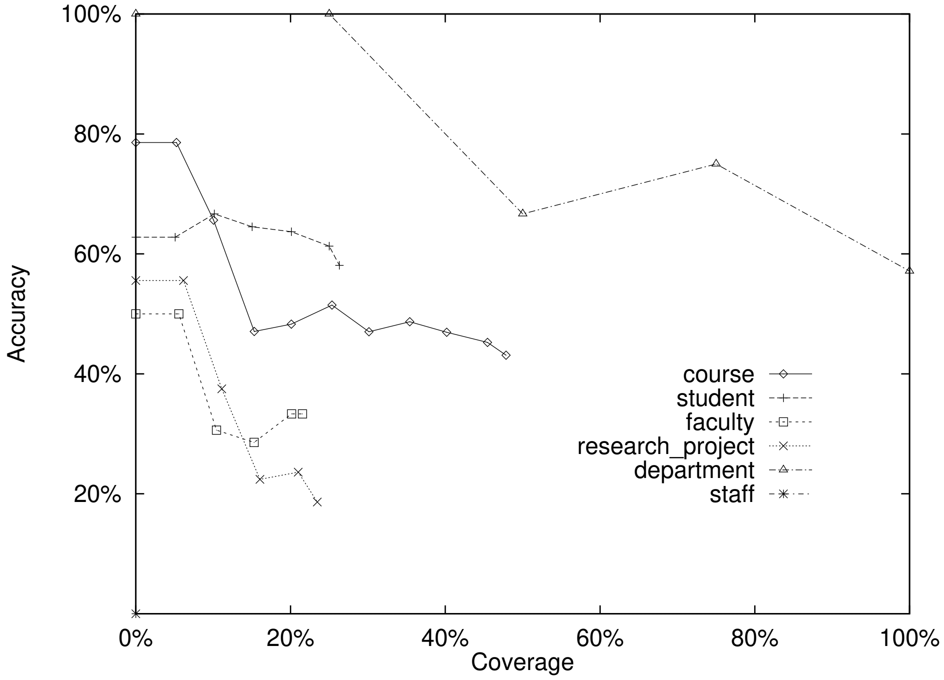

Fig. 5. Accuracy/coverage tradeoff for hyperlink classifiers.

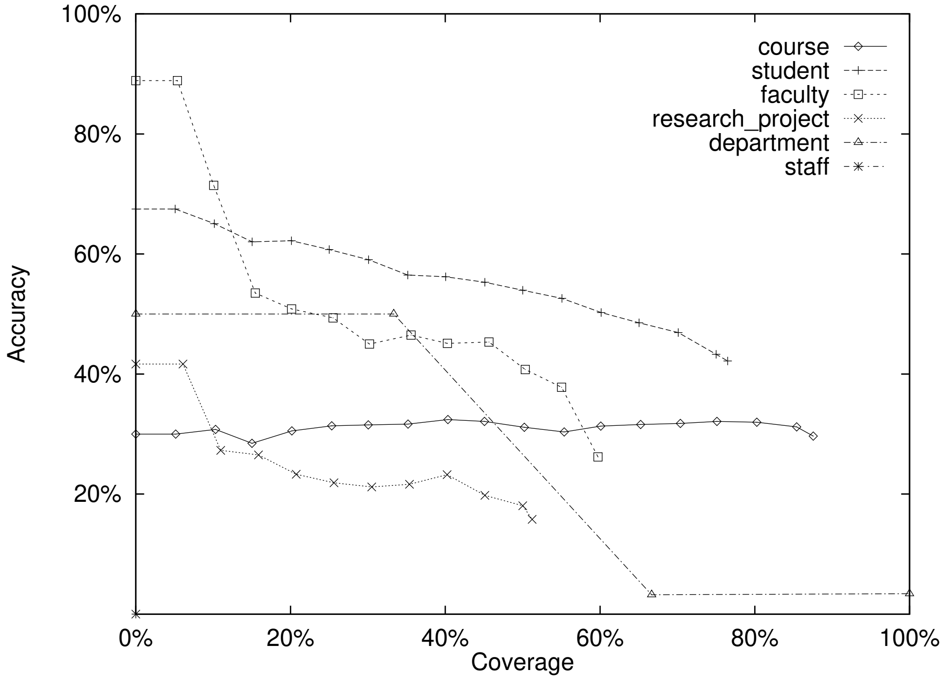

Fig. 6. Accuracy/coverage tradeoff for title/heading classifiers.

coverage of 20%, accuracy reaches a level of 67%.

So far, we have discussed the results only for the full-text classifiers. Figures 5 and 6 show the accuracy/coverage curves for the hyperlink and title/heading classifiers, respectively. As before, these curves show the aggregate results for all four test sets in our cross-validation run.

As we discussed earlier, one of the aspects that distinguishes learning in hypertext from learning in flat-text domains is that hypertext provides multiple, somewhat independent sources of evidence for the meaning of a given piece of text. As we hypothesized, the results in Figures 5 and 6 indicate that these multiple sources of evidence can be potentially combined to make better predictions.

Consider, for example, the accuracy of the department predictions made by the hyperlink classifiers. Whereas the full-text classifiers are only 9% accurate at full coverage, the hyperlink classifiers are 57% accurate. Moreover, the department accuracy/coverage curve for the hyperlink classifiers is uniformly superior to the curve for the full-text classifiers. The reason for this difference in accuracy is that although our data set includes few department *pages* from which to generalize, it includes many *hyperlinks* that point to department pages. Thus the hyperlink classifiers have relatively large samples of data from which to learn the word statistics of hyperlinks that point to department pages, and similarly they have a fairly large number of hyperlinks on which to base their prediction when classifying a page after training.

The title/heading classifiers also illustrate cases in which using a hypertext-based representation for page classification can result in better predictive accuracy than simply using a flat-text representation. The title/heading classifiers' curve for both the faculty and research_project classes, for example, are better than the corresponding curves for the full-text classifiers at coverage levels of 40% and less. One explanation for this result is that titles and headings provide something of a summary of a given page and thus tend to contain words that are highly predictive of the page's class.

5.2 First-Order Text Classification

As noted previously, the hypertext structure of the Web can be thought of as a graph in which Web pages are the nodes of the graph and hyperlinks are the edges. The methods for classifying Web pages that we discussed in the previous sections consider the words in either a single node of the graph or in a set of edges impinging on the same node. However, these methods do not allow us to learn models that take into account such features as the pattern of connectivity around a given page, or the words occurring in neighboring pages. It might be profitable to learn, for example, a rule of the form "A page is a course home page if it contains the words *textbook* and *TA* and is linked to a page that contains the word *assignment*." Rules of this type, that are able to represent general characteristics of a graph, can be concisely represented using a first-order representation. In this section, we consider the task of learning to classify pages using a learner that is able to induce first-order rules.

5.2.1 Approach

The learning algorithm that we use in this section is Quinlan's FOIL algorithm [52,53]. FOIL is a greedy covering algorithm for learning function-free Horn clauses [4]. FOIL induces each Horn clause by beginning with an empty tail and using a hill-climbing search to add literals to the tail until the clause covers only (mostly) positive instances. The evaluation function used for the hill-climbing search is an information-theoretic measure.

The representation we provide to the learning algorithm consists of the following background relations:

- has_*word*(Page): This set of relations indicate which words occur in which pages. Each boolean relation indicates the pages in which the word *word* occurs. A distinct relation is used for each allowed *word* (e.g. has_apple, has_study, etc.). The vocabulary for this set includes stemmed [5] words that have at least 200 occurrences but that do not occur in more than 30% of the training-set pages. These two constraints were selected with the intention of assembling a vocabulary of reasonable size that would likely include the words with the most discrimination power. We had between 592 and 729 of these predicates in each of the cross-validation runs. Since we do not know *a priori* which subsets of pages it will profitable to describe in learned rules, we do not select the vocabulary using mutual information, as we did with our statistical text classifiers.
- link_to(Page, Page): This relation represents the hyperlinks that interconnect the pages in the data set.

We apply FOIL to learn a separate set of clauses for six of the seven classes considered in the previous section [6]. We do not learn a description of the other class, but instead treat it as a default class.

When classifying test instances, we calculate an associated measure of confidence along with each prediction. We calculate these confidence values for two reasons. First, we use them to resolve conflicting predictions from our six

[4] We use the terms *clause* and *rule* interchangeably

[5] *Stemming* refers to the process of heuristically reducing words to their root form. For example the words *compute*, *computers* and *computing* would be stemmed to the root *comput*.

[6] There is a version of FOIL specifically designed for multi-class problems such as ours. We found, however, that the inductive bias of this version is not well suited to our particular task.

independently learned rule sets. Second, we are interested in measuring how the accuracy of our learned rule sets varies as we adjust their coverage.

We use the following procedure to calculate the confidence of each of our predictions. First, we estimate the accuray of each of our learned clauses by calculating an m-estimate [12] of the rule's accuracy over the training examples. The m-estimate of a rule's accuracy is defined as follows:

$$m\text{−estimate accuracy} = \frac{n_c + mp}{n + m}$$

where n_c is the number of instances correctly classified by the rule, n is the total number of instances classified by the rule, p is a prior estimate of the rule's accuracy, and m is a constant called the *equivalent sample size* which determines how heavily p is weighted relative to the observed data. In our experiments, we set $m = 2$ and we set p to the proportion of instances in the training set that belong to the target class. We then use these scores to sort the clauses in order of descending accuracy. [7] To integrate the predictions of our six independently learned classifiers, we use the following procedure:

- If no classifier had a rule that matched the given page, then we predict other with confidence 1.0.
- If only one classifier had a matching rule, then we predict the associated class with confidence corresponding to the rule's score. The other class is predicted with confidence of one minus this score.
- If more than one classifier has a matching rule for the given example, then we predict each class with confidence equal to the score of its best matching rule divided by the total number of classifiers that had matching rules. The other class is predicted with a confidence value that would make the total confidence sum to one.

5.2.2 Experimental Evaluation

For the experiments reported here, we used release 6.4 of FOIL with the default settings. As with the experiments in Section 5.1, we use a four-fold cross-validation methodology. The resulting accuracy/coverage plot for each class is shown in Figure 7. Comparing these results to those in Figure 4, one can see that although the first-order rules generally provide lower coverage than the statistical classifiers, they provide superior accuracy for several classes.

Figure 8 shows three of the rules [8] learned by FOIL in its various cross-

[7] This change does not affect the classifications made by a learned set of clauses. It affects only our confidence associated with each prediction.

[8] Throughout the article, we use a Prolog-like syntax for learned rules. The symbol

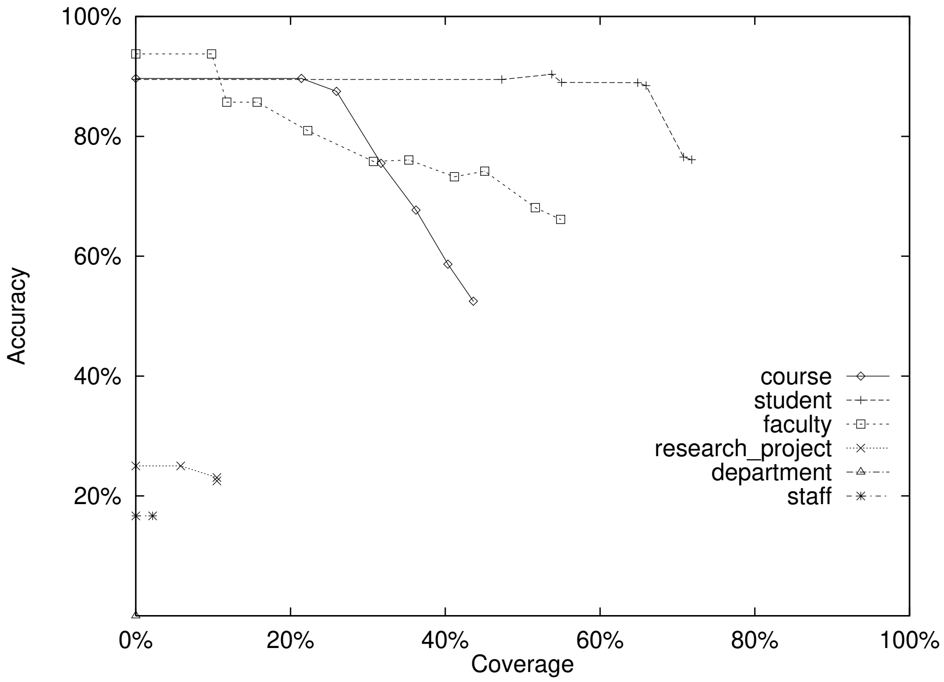

Fig. 7. Accuracy/coverage tradeoff for FOIL page classifiers.

validation runs. The learned rule for course shown here illustrates the power of a first-order representation. This rule classifies a page as the home page for a course if it passes three groups of tests:

(1) The page has the word *instructor*, but doesn't have the word *good*.
(2) The page contains a hyperlink to a page which does not contain any hyperlinks to other pages.
(3) This linked page contains the word *assign*.

The sample rule learned for the student class comes from the cross-validation run leaving pages from the University of Washington out of the training set. Notice that this rule refers to a page (bound to the variable B) that has two common first names on it (*paul* and *jame*, the stemmed version of *james*). This rule (and similar rules learned with the other three training sets) illustrates that FOIL has learned to exploit "student directory" pages in order to identify student home pages. For example, when Washington is the test set, all of the correct applications of the rule bind B to a page entitled "Graduate Students at UW CS&E". Similarly, the faculty rule will not classify a page as faculty unless there is a page containing the stemmed variant of *faculty* that points into the given page.

:- represents the implication operator, with the head of the rule on the left side of the operator and the body on the right side. Constants, such as the names of our ontology classes and relations, start with lowercase letters. Variables start with uppercase letters.

student(A) :- not(has_*data*(A)), not(has_*comment*(A)), link_to(B,A), has_*jame*(B), has_*paul*(B), not(has_*mail*(B)).
`Training Set: 147 Pos, 0 Neg; Test Set: 126 Pos, 5 Neg`

faculty(A) :- has_*professor*(A), has_*ph*(A), link_to(B,A), has_*faculti*(B).
`Training Set: 47 Pos, 0 Neg; Test Set: 18 Pos, 3 Neg`

course(A) :- has_*instructor*(A), not(has_*good*(A)), link_to(A,B), not(link_to(B,_1)), has_*assign*(B).
`Training Set: 31 Pos, 0 Neg; Test Set: 31 Pos, 3 Neg`

Fig. 8. A few of the rules learned by FOIL for classifying pages.

All three of these rules show how Web-page classification is different from ordinary text classification in that neighboring pages may provide strong evidence about the class of a given page. Learning methods which can use this information effectively should perform better than standard techniques in this domain.

5.3 Combining Learners

The previous experiments show that the best representation for page classification depends on the class. This observation suggests that it might be profitable to combine the predictions made by our four classifiers. In this section, we describe and evaluate a simple approach to this task.

5.3.1 Approach

The method that we employ for combining the predictions of our classifiers takes advantage of the fact that each classifier produces a measure of confidence along with each prediction. The method we use is a simple voting scheme that uses confidence values as tie-breakers. That is, given the predictions made by our four classifiers for a given Web page, we predict the class that has a plurality of the votes made by the individual classifiers, if there is one. If no class has a plurality, then we select the class associated with the highest confidence prediction.

In order to ensure that the confidence measures output by our different classifiers are comparable, we calibrate each classifier by inducing a mapping from its output scores to the probability of a prediction being correct. We do this by partitioning the scores produced by each classifier into bins and then measuring the training-set accuracy of the scores that fall into each bin.

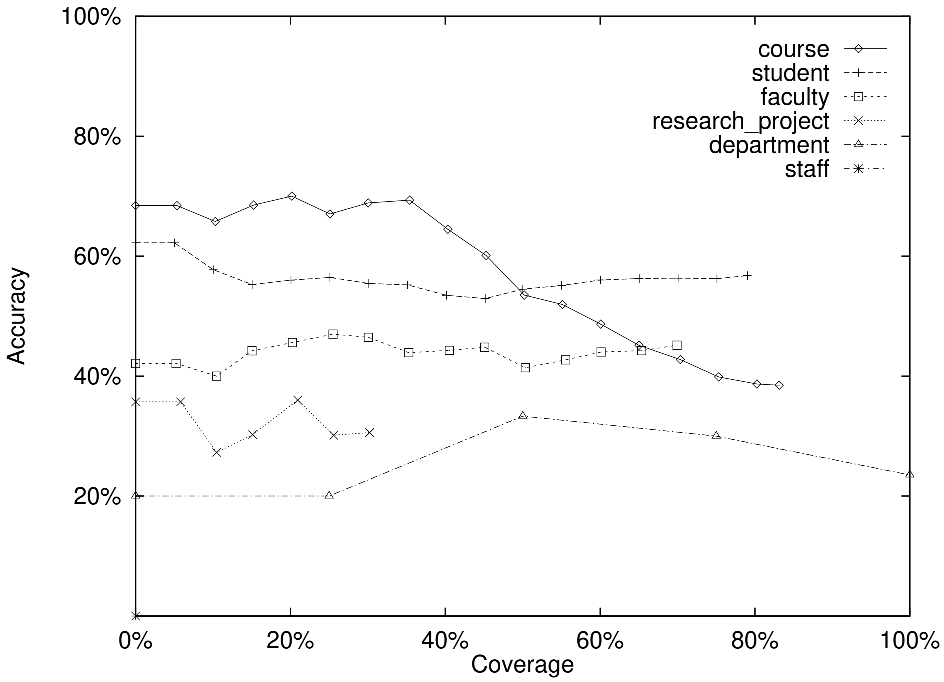

Fig. 9. Accuracy/coverage tradeoff for combined classifiers with vocabulary size of 2000 words.

5.3.2 Experimental Evaluation

Figure 9 shows the accuracy/coverage curves for the voting predictors. By comparing this figure to the accuracy/coverage curves for the full-text classifiers shown in Figure 4 one can see that, in general, more accurate predictions are achieved by considering evidence other than full-text when classifying pages. At high levels of coverage, the voting classifiers are more accurate than the full-text classifiers for the course and department classes. Additionally, the research_project predictions made by the voting classifier are significantly more accurate than the full-text predictions, although the coverage attained by the voting classifier is not as good.

Although Figure 9 indicates that predictive accuracy is helped in some cases by combining multiple classifiers, the results of this experiment are somewhat disappointing. The accuracy/coverage curves for the voting classifiers are not uniformly better than the corresponding curves of the constituent classifiers. Ideally, we would like the accuracy/coverage curve for each class to be as good or better than the *best* counterpart curve among the constituent classifiers.

We believe that the results shown in Figure 9 are disappointing because our method for combining the predictions of multiple classifiers is overly simple. Specifically, we believe that the method fails to accurately map classification scores to estimated accuracies. Interestingly, we have observed that the voting method performs much better when our statistical classifiers are limited to very small vocabularies. Figure 10 shows the accuracy/coverage curves for

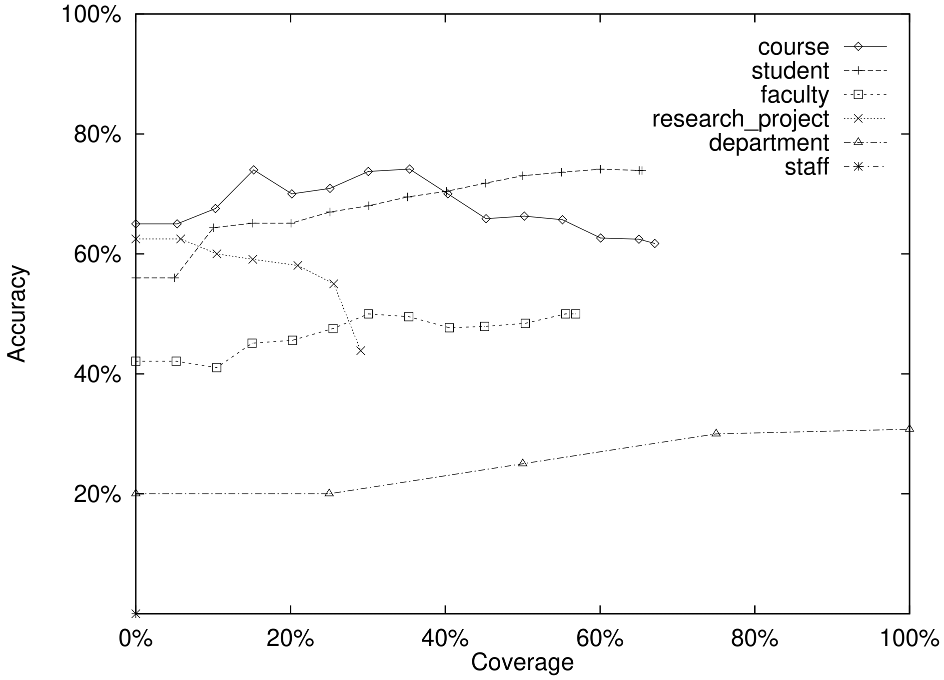

Fig. 10. Accuracy/coverage tradeoff for combined classifiers with vocabulary size of 200 words.

voting when we use statistical classifiers trained with a vocabulary size of 200 words. In comparing this figure to our baseline full-text classifier (Figure 4), one can see that the curves produced by the small-vocabulary voting method are generally superior to the full-text classifier curves. Moreover, the small-vocabulary voting classifiers achieved this result using constituent classifiers that were not as accurate as their 2000-word vocabulary counterparts.

In future work, we plan to consider alternative combining functions that might be better able to exploit the specialized areas of expertise exhibited by our individual classifiers.

5.4 Identifying Multi-Page Segments

As discussed in Section 3, our representational assumption is that each class instance in the knowledge base corresponds to some contiguous segment of hypertext on the Web. This allows, for example, that a particular student might be represented on the Web by a single Web page, or by a cluster of interlinked Web pages centered around their home page.

In the experiments reported thus far, we have effectively made a simpler assumption: that each instance is represented by a single Web page. In fact, in labeling our training data, we encountered a variety of students (and instances of other ontology classes) that were described by several interlinked Web pages rather than a single page. In these cases we hand labeled the primary home page as student, and labeled any interlinked pages associated with the same student as other.

To remove this simplifying assumption we must develop methods for identifying sets of interlinked pages that represent a single knowledge base instance. In this section we present a set of hand-written heuristics that identify groups of related pages and also identify the “primary” home page in the group. We show here that classification accuracy in the previous sections is significantly improved when these heuristics are used to group pages and to automatically assign the label other to non-primary pages, to fit the assumption we made while hand labeling the data.

5.4.1 Approach

Consider the Web pages of a prototypical faculty member. She might have a main page (`http://www.my.edu/user/jdoe/index.html`), a page listing her publications (`http://www.my.edu/user/jdoe/pubs.html`), and a page describing her research interests (`http://www.my.edu/user/jdoe/work/research.html`). Our working assumption about entity-Web relationships indicates that we should recognize that these pages correspond to a single entity, identify the best representative page for that entity, classify that page as a faculty, and classify the rest of the pages as other. We accomplish this by solving two subtasks: grouping related pages together, and identifying the most representative page of a group.

Spertus [64] identifies regularities in URL structure and naming, and presents several heuristics for discovering page groupings and identifying representative home page. We use a similar, slightly expanded, approach. Although one could imagine trying to learn these heuristics from examples, in the following experiment we have instead provided these rules by hand.

The most obvious groupings that can be extracted from a URL are based on directory structure prefixes. Key directory components of a URL indicate a logical grouping of Web pages into an entity. For example, given the URL `http://www.my.edu/user/jdoe/research.html`, we can deduce the existence of an entity corresponding to the URL prefix `http://www.my.edu/user/jdoe/`, because the keyword `/user/` in the penultimate directory position typically indicates the presence of a person entity in the directory space denoted by `jdoe`. Other typical penultimate prefix markers are `/faculty/`, `/people/`, `/home/`, and `/projects/`. Three ultimate prefix markers (in UNIX-style globbing pattern) are `/cs???/`, `/www/` and `/~*/`, the first being a typical indicator of a course, and the last being a typical indicator of the username of a person or organization. Our algorithm groups URLs by their longest directory prefix that matches one of these given patterns. In the event that no pattern matches, the entire directory prefix is used for the grouping. In our example above, the three URLs would each have the entity prefix as `http://www.my.edu/user/jdoe/`, and thus would be grouped together.

Applying these grouping heuristics results in sets of Web pages that are conjectured to represent a single ontology entity. From these sets, we identify the single primary page that is most representative of that entity. Usually this corresponds to the "home page" of the entity. Thus, we take any page that has the filename pattern "`index.html`", "`home.html`", "`homepage.html`", or "`cs???.html`" and label it the primary page. Additionally, any page in which the complete URL is the directory prefix, (for example, the URL `http://www.my.edu/user/jdoe/`) or one in which the filename matches the directory above it (as in `http://www.my.edu/user/jdoe/jdoe.html`) is also identified as a primary page. All pages that do not match any of these patterns in a group, are classified automatically as other. In the event that no page in a group matches any of these heuristics, the page with the highest (non-other) classification confidence is labeled the primary page. In our example, `http://www.my.edu/user/jdoe/index.html` would be classified as faculty (assuming our classifier was correct), and the other pages would be classified as other regardless of the classifier prediction. A precise definition of the algorithm used is given in Appendix C.

5.4.2 Experimental Evaluation

The impact of using the URL heuristics with the original full-text page classifier is summarized in Figure 11. Comparing these curves to Figure 4 one can see the striking increase in accuracy for any given level of coverage across all classes. Also note some degradation in total coverage. This occurs because some pages that were previously correctly classified have been misidentified as being "secondary" pages.

5.5 Section Summary

This section focused on the task of recognizing class instances by Web page classification. We showed that, because hypertext provides much redundant information, Web pages can be classified using several sources of information: the full text of pages, the text in titles and headings, the text associated with hyperlinks, text in neighboring pages, and the file organization represented in URLs. Our experiments suggest that none of these approaches alone is sufficient for recognizing instances of ontology classes with high accuracy. In the experiments described in Section 2, we used both full-text classifiers and

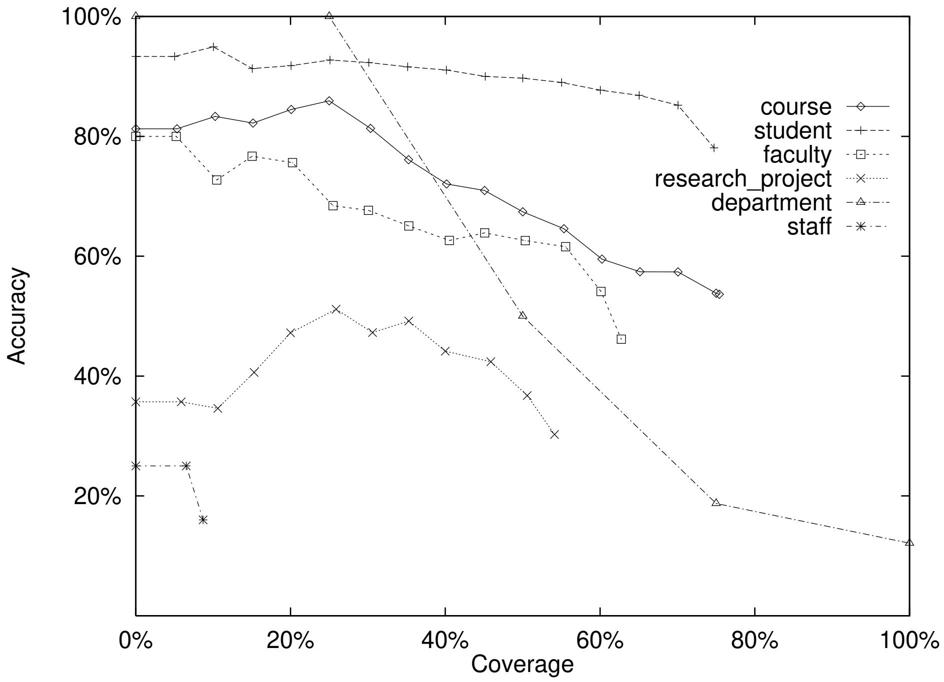

Fig. 11. Accuracy/coverage tradeoff for the full-text classifier after the application of URL heuristics.

URL heuristics. We also showed in this section that one promising line of research is to combine the predictions of multiple classifiers that use different sources of evidence.

6 Learning to Recognize Relation Instances

In the previous section we discussed the task of learning to extract instances of ontology classes from the Web. Our approach to this task assumed that the class instances of interest are represented by whole Web pages or by clusters of Web pages. In this section, we discuss the task of learning to recognize *relations* of interest that exist among extracted class instances. The hypothesis underlying our approach is that relations among class instances are often represented by *hyperlink paths* in the Web. Thus, the task of learning to recognize instances of such relations involves inducing rules that characterize the prototypical paths of the relation.

For example, an instance of the instructors_of relation might be represented by a hyperlink directly from the home page of a course to the home page of the instructor, as described by the following rule:

instructors_of(A, B) :- course(A), person(B), link_to(A, B).

Here, the variables A and B represent Web pages, the literals course(B) and person(A) represent the predicted classifications of the pages, and the literal link_to(A, B) tests for the existence of a hyperlink from page A to page B.

6.1 Problem Representation

Because this task involves discovering hyperlink paths of unknown and variable size, we employ a learning method that uses a first-order representation for its learned rules. Specifically, the algorithm we have developed for this task is based on the FOIL algorithm [52,53] which we used for page classification in Section 5.2. We discuss our algorithm in more detail below.

The problem representation we use for this relation learning tasks consists of the following background relations:

- *class*(Page) : For each *class* in the set of page classes considered in Section 5, the *class* relation lists the pages that represent instances of *class*. For pages in the training set, the instances of these relations are determined using the actual classes of the pages. For pages in the test set, however, we use the predicted page classes given by the classifiers discussed in Section 5. Since the WEB→KB system has access only to predicted page classes, our test set conditions are representative of those the system faces.
- link_to(Hyperlink, Page, Page) : This relation represents Web hyperlinks. For a given hyperlink, the first argument of the relation specifies an identifier for the hyperlink, the second argument specifies the page in which the hyperlink is located, and the third argument indicates the page to which the hyperlink points.
- has_*word*(Hyperlink) : This set of relations indicates the words that are found in the anchor (i.e., underlined) text of each hyperlink. The vocabulary for this set of relations includes words that occur at least n times (we set $n = 3$ in our experiments) in the hyperlinks of the training set. Note that whereas the has_*word* relations used in Section 5.2 describes Web pages, the set used here characterizes hyperlinks.
- all_words_capitalized(Hyperlink) : The instances of this relation are those hyperlinks in which all of the words in the anchor text start with a capital letter.
- has_alphanumeric_word(Hyperlink) : The instances of this relation are those hyperlinks which contain a word with both alphabetic and numeric characters (e.g., I teach CS760).
- has_neighborhood_*word*(Hyperlink) : This set of relations indicates the words that are found in the "neighborhood" of each hyperlink. The neighborhood of a hyperlink includes words in a single paragraph, list item, table entry, title or heading in which the hyperlink is contained. The vocabulary for this set of relations includes the 200 most frequently occurring words in

each training set, except for words on a stoplist.

We learn definitions for the following target relations from the data set described in Section 4: members_of_project(Page, Page), instructors_of_course(Page, Page), and department_of_person(Page, Page). In addition to the positive instances for these relations, our training sets include approximately 300,000 negative examples. We form the set of negative training instances for each target relation by enumerating each pair of non-other pages from the same university that is not a positive instance of the target relation. This selection criterion results in a sample of negative instances that is biased toward connected pairs of pages and is small enough to allow reasonable learning times. For the department_of_person relation, we augment the negative instances with each person–department pair which is not a positive instance. These additional negative instances preclude the learner from learning the trivial (and wrong) rule that every person is a member of every department.

6.2 Learning Methods

As stated above, the algorithm we use for learning relation rules is similar to FOIL in that it uses a greedy covering approach to learn a set of Horn clauses. The primary differences between our method and FOIL are twofold. First, unlike FOIL our method does not simply use hill-climbing when searching for the next clause to add to a concept definition. Second, our method uses a different evaluation function for this search process. We discuss each of these differences in turn.

As described in Section 5.2, FOIL constructs clauses using a hill-climbing search through a space of candidate literals. We have found that, for our relation-learning tasks, such a hill-climbing strategy is unable to learn rules for paths consisting of more than one hyperlink. The search process that our method employs instead consists of two phases. In the first phase, the "path" part of the clause is learned, and in the second phase, additional literals are added to the clause using a hill-climbing search.

Our algorithm for constructing the path part of a clause is a variant of Richards and Mooney's *relational pathfinding* method [55]. This method is designed to alleviate the basic weakness of hill-climbing search, namely that to learn good definitions it is often necessary to take a step in the search space which does not exhibit any immediate gain. The basic idea underlying relational pathfinding is that a relational problem domain can be thought of as a directed graph in which the nodes are the domain's constants and the edges correspond to relations which hold among constants. The relational-pathfinding algorithm tries to find a small number of prototypical paths in this graph that characterize

Input: training set of negative and uncovered positive instances

1. for each uncovered positive instance
2. find a path (up to bounded length) using the background relations
3. select the most common path prototype for which clause search hasn't yet failed
4. generalize the path into an initial clause
5. do hill-climbing to refine the clause
6. if hill-climbing fails to find an acceptable clause, backtrack to step 3.

Return: learned clause

Fig. 12. The procedure for learning a clause in our deterministic variant of relational pathfinding.

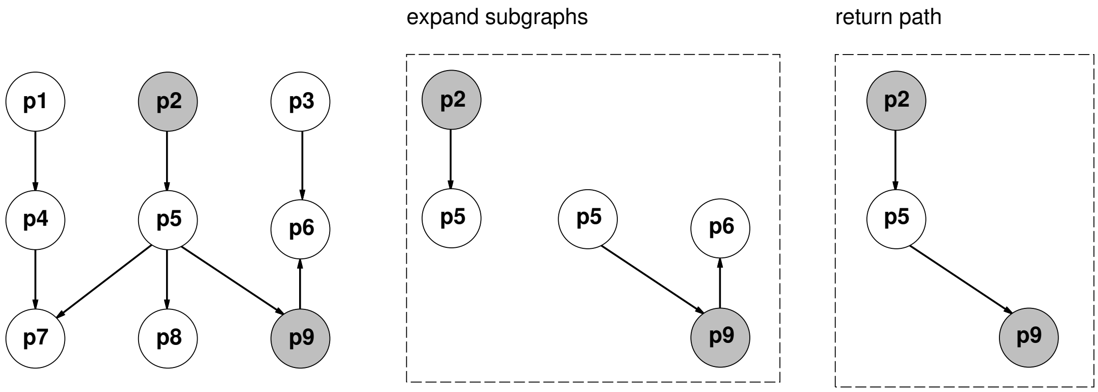

Fig. 13. Finding a path in the background relations. On the left is shown a graph of constants linked by a single binary relation. This graph can be thought of as representing Web pages connected by hyperlinks. Suppose the pair ⟨p2, p9⟩ is an uncovered positive instance. Pathfinding proceeds by expanding the subgraphs around the two constants until an intersection is detected, and then returning the path that links the two constants.

the instances of the target relation.

Figure 12 provides an overview of our pathfinding procedure for learning a single clause. This procedure is iterated until a complete definition has been learned. The first step in the method is to find the shortest path of a bounded length (when one exists) for each positive instance (of the target relation) that has not been covered by a previously learned clause. This process, illustrated in Figure 13 involves expanding a subgraph around each of the constants in the instance. Each subgraph is expanded by finding all constants which can be reached using an instance of one of the background relations to connect to a constant at the frontier of the subgraph.

After finding such a path for each uncovered positive instance, the most com-

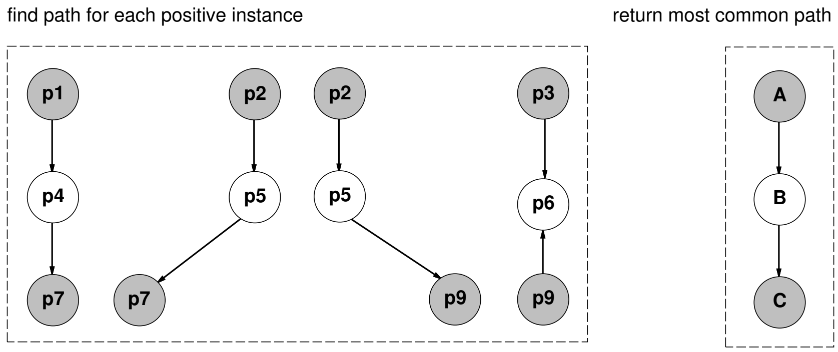

Fig. 14. Finding the most common path for a set of positive instances. Given the graph shown in Figure 13, suppose that the positive instances are ⟨p1, p7⟩, ⟨p2, p7⟩, ⟨p2, p9⟩, and ⟨p3, p9⟩. Our algorithm finds the shortest path for each instance and then returns the most common path prototype. In this example the first three instances have the same path prototype, whereas the instance ⟨p3, p9⟩ has different one (notice the direction of the hyperlinks). This path prototype is converted into an initial clause.

mon path prototype is used for the initial clause. [9] A path prototype specifies the number of hyperlinks in the path and their directions, but it does not reference the particular pages and hyperlinks in any particular instance. The notion of the most common path prototype is illustrated in Figure 14. The initial clause is formed by replacing each constant in the path with a unique variable. This clause is then further refined by a simple hill-climbing search, such as that used in FOIL. If the hill-climbing search fails to find an acceptable clause, then the procedure backtracks by removing the last selected path prototype from the list of candidates and then trying the next most common prototype.

We further bias the search for clauses by initializing each one with the classes of the pair of pages in the relation. For example, when learning clauses for the target relation members_of_project(A, B), we initialize the tail of each clause with the literal research_project(A) and person(B). This bias takes advantage of domain knowledge which is present in the ontology given to the WEB→KB system.

The second difference between our relation-learning algorithm and FOIL is that whereas FOIL uses an information-theoretic measure to guide its hill-climbing search, our method, like Džeroski and Bratko's m-FOIL [19], uses m-estimates of a clause's error to guide its construction. We have found that using this evaluation function causes the algorithm to learn fewer, more general clauses

[9] If the method is constrained from learning recursive definitions, the path for each positive instance needs to be found only once since it will not change as clauses are added for the target relation. In this case, before learning each new clause the algorithm needs only to update counts indicating the number of instances covered by each path prototype.

than when FOIL's information gain measure is used.

6.3 Experimental Evaluation

We evaluate our approach to learning relation rules using the data and four-fold cross-validation methodology described in Section 4. On each iteration, we learn the target relations using training instances from three of the universities in our data set, and test learned clauses using instances from the fourth university.

Figure 15 shows a learned clause for each of the instructors_of, department_of, and members_of_project relations. On average, there were 7.3, 3.8, and 6.5 clauses learned for these target concepts respectively. Along with each rule, we show how well the rule classified test-set instances. Each of these rules was learned on more than one of the training sets, therefore the test-set statistics represent aggregates over the four test sets.

The rules learned for the instructors_of relation are the simplest among the three target relations. The learned rule shown for this relation, for example, matches cases in which a course page has a hyperlink pointing to a person page. The rule shown for the members_of_project relation is more interesting. It describes members_of_project instances in which the project's home page points to an intermediate page which points to personal home pages. The hyperlink from the project page to the intermediate page must have the word "people" near it. This rule covers cases in which the members of a research project are listed on a subsidiary "members" page instead of on the home page of the project. The rule shown for the department_of relation involves a three-hyperlink path that links a department home page to a personal home page. The rule requires that the word "graduate" occur near the second hyperlink in the path. In this case, the algorithm has learned to exploit the fact that departments often have a page that serves as a graduate student directory, and that any student whose home page is pointed to by this directory is a member of the department.

Along with each of our predicted relation instances, we calculate an associated confidence in the prediction. We can then vary the coverage of our learned rule sets by varying a threshold on these confidence values. We calculate the confidence of each prediction by considering where most of the uncertainty in the prediction lies: in the page classifications that are tested by each learned clause. The confidence measure for a predicted relation instance is simply the product of the confidence measures for the page classifications that factor into the relation prediction.

Using these confidence measures, Figure 16 shows the test-set accuracy/coverage instructors_of(A,B) :- course(A), person(B), link_to(C,B,A).

`Test Set: 133 Pos, 5 Neg`

department_of(A,B) :- person(A), department(B), link_to(C,D,A), link_to(E,F,D), link_to(G,B,F), has_neighborhood_word_*graduate*(E).

`Test Set: 371 Pos, 4 Neg`

members_of_project(A,B) :- research_project(A), person(B), link_to(C,A,D), link_to(E,D,B), has_neighborhood_word_*people*(C).

`Test Set: 18 Pos, 0 Neg`

Fig. 15. A few of the rules learned for recognizing relation instances.

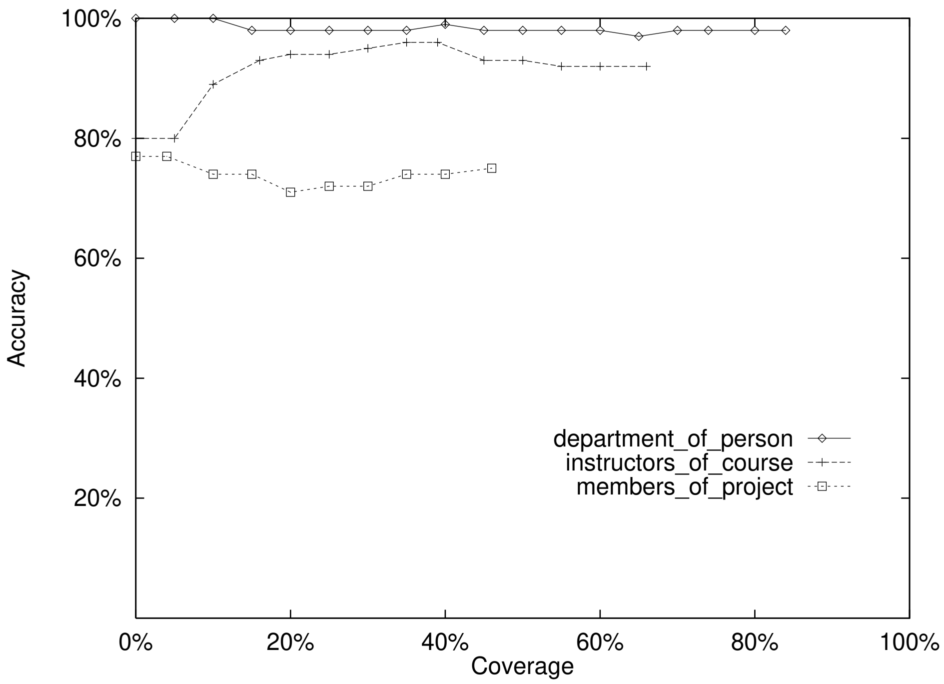

Fig. 16. Accuracy/coverage tradeoff for learned relation rules.

curves for the three target relations. The accuracy levels of all three rule sets are fairly high. The members_of_project rules are better than 70% accurate at coverage levels of up to about 46%. The instructors_of rules are over 80% accurate at coverage levels of 66% and above. The department_of rules are at least 97% accurate at coverage levels of up to 84%. The limited coverage levels of the learned rules is due primarily to the limited coverage of our page classifiers. Note that all of the learned rules include literals which test predicted page classifications. As Figure 11 shows, the coverage exhibited by our page classifiers is below 80% for most classes.

7 Learning to Extract Text Fields

In some cases, the information we want to extract will not be represented by Web pages or relations among pages, but by small fragments of text embedded in pages. For example, given a personal home page, we might be interested in extracting the person's name. This type of task is commonly called *information extraction.* This section discusses our approach to learning rules for such information extraction tasks.

7.1 Approach

We have developed an information extraction learning algorithm called SRV for "Sequence Rules with Validation." SRV is a first-order learner in the spirit of FOIL. It shares FOIL's top-down approach and gain metric, but is designed with the information extraction problem in mind. Consequently, it is limited to a few pre-defined predicates, and it encompasses search heuristics specific to the information extraction problem. Input to SRV is a set of pages, labeled to identify instances of the field we want to extract, and a set of features defined over tokens. Output is a set of information extraction rules. The extraction process involves examining every possible text fragment of appropriate size to see whether it matches any of the rules.

As in FOIL, "growing" a rule in SRV means hill-climbing through a space of possible literals, at each step adding a literal that matches as many positive examples as possible while excluding a large number of previously covered negative examples. When a rule is deemed good enough (either it covers only positive examples, or further specialization is judged to be unproductive), all positive examples matching it are removed from the training set, and the process is repeated. In our particular domain, a positive example is a labeled text fragment—a sequence of tokens—in one of our training documents; a negative example is any unlabeled token sequence having the same size as some positive example. During training we assess the goodness of a literal using all such negative examples.

The representation used by our rule learner attempts to express the salient characteristics of positive examples mainly in terms of the individual tokens contained within them and surrounding them. SRV is given as input a set of features defined over individual tokens. These features come in two varieties: *simple* features map tokens to arbitrary discrete (but typically Boolean) values; *relational* features map tokens to other tokens in the same document. An example simple feature is capitalized. An example relational feature is next_token which, given a token, returns the token immediately following it.

Let $F = w_i, \cdots, w_j$ be a fragment of text, an unbroken sequence of tokens (w_i) from some document. At each step in rule growth, a literal is generated from one of four templates:

- length(Relop, N): Constrains the length of F. Relop is one of $\{<, >, =\}$ and N is an integer. For example, the literal length(<, 3) means $|F| < 3$, where $|F|$ is the number of tokens in F.[10]
- some(Var, Path, Feat, Value): Posits a feature-value test for some token in the sequence. Var is a variable, Path is a list of relational features, Feat is a simple feature and Value is a legal value of Feat. For example, the literal some(B, [ ], capitalized, true) asserts that F contains a capitalized token and binds this token to the variable B. Each distinct variable in a rule must bind to a distinct token in a matching fragment. In logical terms, if in some other literal in the current rule SRV has introduced the variable A and now makes the above assertion, there is an implicit assertion of inequality between A and B:
  $\exists A, B \in F. A \neq B \wedge \mathit{capitalized}(B) = \mathit{true}$
  The Path argument, which is empty in the above example, is used to exploit relational structure in the domain (see below).
- position(Var, From, Relop, N): Constrains the position of a token bound by a some-literal in the current rule. From is either start or end, Relop is one of $\{<, >, =\}$, and N is an integer. For example, the literal position(A, start, <, 2) asserts that the token bound to A is either the first or second in F. In logic, we might write this $\exists A \in F. \mathit{index}(A) - \mathit{index}(\mathit{start}(F)) < 2 \wedge \cdots$, where the ellipsis ($\cdots$) stands for whatever other assertions are made about A in the same rule.
- relpos(Var1, Var2, Relop, N): Constrains the relative position an ordering of two tokens bound by different variables in the same rule. Relop is one of $\{<, >, =\}$, and N is an integer. For example, the literal relpos(A, B, =, 1) means token B immediately follows token A. In logic:
  $\exists A, B \in F. \mathit{index}(B) - \mathit{index}(A) = 1 \wedge \cdots$.

Like FOIL, SRV can exploit relational structure in the domain. For SRV, the only possible form of relational structure is that relating tokens to each other. The most obvious example is the successor relation, which connects adjacent tokens, but more interesting kinds of structure can be exploited, such as syntactic structure. The Path argument to the some-predicate takes a list of relational features which posits a relationship between the token bound to the variable (which is in F) and the token reached by composing the relational features (which need not be in F). The feature-value test is applied to the "indirect" token. For example, the literal some(A, [next_token next_token], numeric, true) might be rendered in logic as $\exists A \in F. \mathit{numeric}(\mathit{next\_token}(\mathit{next\_token}(A))) = \mathit{true}$.

[10] Note that all literals added to a rule implicitly refer to some fragment F, so we omit the fragment variable from SRV literals for the sake of conciseness.

```
ownername(Fragment) :- some(B, [ ], in_title, true),
                       length(<, 3),
                       some(B, [prev_token], word, "gmt"),
                       some(A, [ ], longp, true),
                       some(B, [ ], word, unknown),
                       some(B, [ ], quadrupletonp, false)
```

Fig. 17. An extraction rule for name of home page owner. The English rendering of this rule is, "a sequence of two tokens, one of which (A) is in a HTML title field and longer than four characters, the other of which (B) is preceded by the token gmt, is unknown from training, and is not a four-character token." This is a high-accuracy rule, achieving 10 correct out of 12 matched on a validation set.

Initially, the learner may only use paths of length zero or one; whenever a non-empty path is used in a some predicate, the system makes longer paths available. In this way, the computational expense that this facility entails is kept under control.

7.2 *Experimental Evaluation*

As in the previous experiments, we followed the leave-one-university-out methodology, repeatedly holding the pages belonging to one of the four universities out for testing and training on the remaining three. The data set for the present experiment consists of all person pages in the data set. The unit of measurement in this experiment is an individual page. If Srv's most confident prediction on a page corresponds exactly to some instance of the page owner's name, or if it makes no prediction for a page containing no name, its behavior is counted as correct. Otherwise, it is counted as an error.

```
Last-Modified: Wednesday, 26-Jun-96 01:37:46 GMT

<title> Bruce Randall  Donald</title>

<h1>
<img src="ftp://ftp.cs.cornell.edu/pub/brd/images/brd.gif">
<p>
Bruce Randall Donald<br>
Associate Professor<br>
```

Fig. 18. An example HTML fragment which the rule in Figure 17 matches. In this case, the fragment `Bruce Randall` in the title is extracted. Note that this is an erroneous prediction since it misses the last name of the person.

Figures 17 and 18 show a learned rule and its application to a test case. Figure 19 shows the accuracy-coverage curve for Srv on the name-extraction task. Under the criteria described above, it achieves 65.1% accuracy when all pages are processed. A full 16% of the files did not contain their owners'

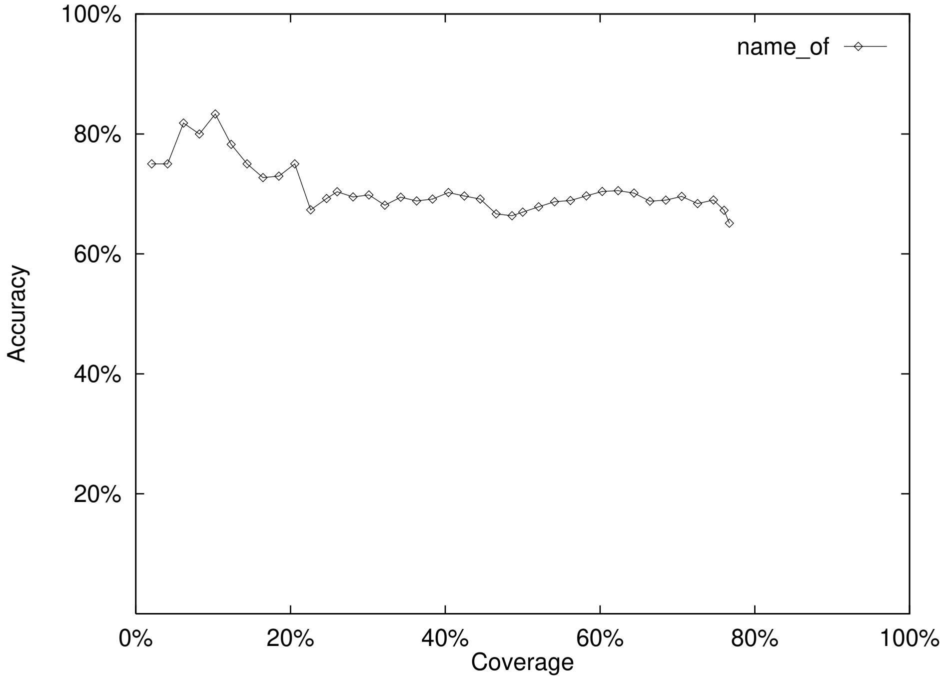

Fig. 19. Accuracy/coverage tradeoff using Srv for name extraction. A prediction on a file that does not contain a name is counted as an error.

names, however, and a large part of the learner's error is because of spurious predictions over these files. If we consider only the pages containing names, Srv's performance is 77.4%.

8 Related Work

There are several significant bodies of research that are related to the tasks and methods discussed in this article. In this section we briefly review the main areas of related work.

8.1 Document Classification

Our work is related to research in document classification, such as that reported at recent Text REtrieval Conferences (TREC) [3,4]. A wide variety of methods have been applied to the document-classification task.

The TFIDF approach to information retrieval is the basis for the Rocchio classification algorithm which has become a standard baseline algorithm for text classification [8,15,34]. Its "word-vector" approach involves describing classes with a vector of weights, where each weight indicates how important the corresponding word is to the class. This representation has been used with many different learning algorithms, including memory based reasoning [40], neural networks [49,58], linear discriminant analysis [58], logistic regression [58], Widrow-Hoff and the exponentiated gradient (EG) algorithm [36].

Another useful line of research in text classification comes from basic ideas in probability and information theory. Bayes Rule has been the starting point for a number of classification algorithms [5,6,35,37,45,49], and the Minimum Description Length principle has been used as the basis of an algorithm as well [34].

Another line of research has been to use symbolic learning methods for text classification. Numerous studies have used algorithms such as decision trees, Swap-1, Ripper and Charade can be found in [5,6,8,13,36,37,45,46,49,65]. These studies indicate that these algorithms are quite competitive with statistical-based methods.

8.2 Information Extraction

The problem that we are addressing is related to the traditional information extraction task, such as the research done in the Message Understanding (MUC) [1,2] community. The work in the MUC community has considered problems such as extracting symbolic descriptions of terrorist attacks from news articles, constructing case frames that indicate fields such as the perpetrator, victim, etc. One key difference between this work and the research reported here is that we are concerned with extracting information from *hypertext*, whereas the MUC work has focused on ordinary *flat text*. In addition, our approach relies heavily on machine learning methods that can be trained to extract information, whereas most early work in the MUC community relied on hand-crafted methods for extracting information.

Recently, the problem of using machine-learning methods to induce information-extraction routines has received more attention. PALKA [30] and AutoSlog [57] are machine learning systems which learn extraction patterns from collections of parsed documents that have been annotated to identify fragments of interest. These patterns are then reviewed and manually installed into a larger information extraction system. AutoSlog-TS [56] removes the requirement that documents be annotated.

CRYSTAL [61] and RAPIER [11] both demonstrate that machine learning techniques can be used to learn rules that perform extraction autonomously. CRYSTAL is a covering algorithm which takes parsed, annotated sentences as input and produces rules for extracting from novel sentences. Rapier uses ideas from relational learning and relaxes somewhat the reliance on syntactic preprocessing. Starting with maximally specific extraction patterns, both systems learn by dropping constraints and merging patterns. This contrasts with the general-to-specific approach introduced here.

Several researchers have explored the problem of text extraction from the Web and other Internet sources. One example is ILA [50], a system designed to learn the semantics of the human-readable output of online databases by comparing it with information whose is already known. Shopbot [18], a bargain hunting agent, is designed to learn patterns from HTML to support the extraction of pricing information from online commerical catalogs. Shopbot is one solution to the general problem of "wrapper induction" [33,47,26], learning extraction patterns for highly regular sources. At the same time, ideas that have proven useful for general text have also been shown to work well for Web pages. Webfoot [62] is a modification of CRYSTAL in which parsed sentence fragments are replaced by segments of HTML. WHISK [63] combines the capabilities of CRYSTAL and Webfoot in a system that is able to learn extraction patterns for semi-structured or free text.

8.3 Extracting Semantic Information from Hypertext

Several other research groups have considered the semantic information that can be automatically inferred and extracted from hypertext. Spertus [64] presents a set of heuristics that relate hypertext conventions to semantic relationships. Specifically, she considers relationships that can often be inferred from hyperlink structure, file system organization, and HTML page structure.

Monge and Elkan [44] have developed a system that finds the Web page for a paper given a bibliographic citation to it. Part of the task performed by this system is to find the personal home page and the publications page of an author starting from the home page of the person's institution. For this task, Monge and Elkan use search-control rules which are somewhat similar to the relation-recognition rules we learned in Section 6. Their rules look for certain keywords in hyperlinks to decide which ones to follow in the search. Whereas their rules are hand-coded for a specific task, our work considers the problem of learning such rules for arbitrary relations.

Pirolli *et al.* [51] consider the task of classifying pages into functional categories such as *head*, *index* and *reference*. They characterize the classes using features such as file size, number of incoming and outgoing hyperlinks, average depth of children pages in the hyperlink graph, etc. Whereas our work has not directly involved learning functional classes of pages, we have observed that our first-order learners for both page and relation classification often implicitly learn such functional categories. Recall, for example, that our learned first-order rules for recognizing student pages prominently exploited the class of *person* *index* pages. The features we use also differ somewhat from those of Pirolli *et al.*, but common to both approaches is the central importance of vector-based text similarity and hyperlink connectivity.

8.4 Extracting Knowledge Bases from the Web

Other groups have worked on extracting propositional knowledge-base information from the Web. Luke *et al.* [39] have proposed an extension to HTML called SHOE whereby Web page authors can encode ontological information on their pages. The have also developed a system, Expose, that extracts SHOE-encoded information from Web pages, and stores it in a local knowledge base. Their hope is that a library of standard ontologies will come into common usage, enabling agents such as Expose to learn the information encoded on the Web.

The START Information Server [29] provides a natural language interface to a knowledge base collected from the Web. The knowledge base contains meta-information about the content of the Web, so that a query to START returns relevant hypertext segments. START builds its knowledge base by discovering mostly manually added natural language annotations on Web pages.

The most significant recent development in this area is the advent of Extensible Markup Language (XML) [10]. Whereas HTML is designed to describe the layout of information in a page, XML can be used to describe information about the *contents* of the page. As with SHOE, Web page authors can use XML to encode ontological information about their pages. Since XML is a World Wide Web Consortium standard, however, it is sure to be widely used. We believe that methods for annotating the contents of Web pages, such as SHOE and XML, can assist with the task of extracting knowledge bases from the Web, but do not obviate the need for our WEB→KB approach. There are two notable limitations of approaches such as SHOE and XML. First, they are of no use when Web page authors do not employ them. Second, they presuppose a universal ontology. That is, since individual Web page authors are responsible for annotating Web pages, the success of these approaches hinges on the extent to which authors employ standard, shared ontologies in a consistent manner. Moreover, ontological decisions are largely in the hands of Web page authors in this approach. There may be cases where the ontological categories used to describe a given Web page are not appropriate or relevant categories for the tasks to which an extracted knowledge base will be applied. In the WEB→KB approach, on the other hand, these ontological decisions can be made by the users of the system. One interesting way in the XML and WEB→KB approaches can potentially be combined, is by exploiting XML-annotated pages as pre-labeled training data. That is, WEB→KB could learn to predict the XML annotations associated with a page, using the non-XML elements of the page as input features. We plan to explore this issue in future research.

8.5 Web Agents

The WEB→KB system described here is an example of a Web agent that browses the Web, extracting information as it goes. Many other Web agents have been developed over the past few years, including several that involve some form of learning. However, the vast majority of these systems use learning to improve their ability to retrieve text information, rather that to extract computer-understandable information. For example, Joachims *et al.* [28] describe a Web agent called WebWatcher that serves as a tour guide for users browsing the Web. WebWatcher learns to suggest appropriate hyperlinks given users' interests, based on the hyperlinks followed by previous users with similar interests. As such, it involves learning to classify hyperlinks – a task similar to the work reported here on learning to extract relational information. A system with a similar goal is Letizia [38], which learns the interests of a single user, in contrast to WebWatcher which learns from a community of users. *Syskill and Webert* [49] offers a more restricted way of browsing than WebWatcher and Letizia. Starting from a manually constructed index page for a particular topic, the user can rate hyperlinks off this page. The system uses the ratings to learn a user specific topic profile that can be used to suggest unexplored hyperlinks on the page. Syskill and Webert can also use search engines like LYCOS to retrieve pages by turning the topic profile into a query. *Lira* [7] works in an off-line setting. A general model of one user's interest is learned by asking the user to rate pages. Lira uses the model to browse the Web off-line and returns a set of pages that match the user's interest. One related system that is closer in spirit to our work is Shakes *et al.*'s [59] Ahoy system, which attempts to locate the home page of a person, given information such as the person's name, organizational affiliation etc. Ahoy uses knowledge of home page placement conventions to search for personal home pages, and in fact learns these conventions from experience.

9 Conclusions and Future Work

We began this article with a goal and a thesis. The goal is to automatically create a large knowledge base whose content mirrors that of the World Wide. The thesis is that one can automatically create knowledge bases from the Web by first using machine learning algorithms to create information extraction methods for each of the desired types of knowledge, and then applying these methods to extract probabilistic, symbolic statements directly from Web hypertext.

This article provides support for our thesis by proposing and testing a variety of machine learning algorithms for information extraction, and by describing the WEB→KB system that incorporates the learned information extractors to browse Web sites and populate a knowledge base. As shown in Section 2 and elsewhere, our system has achieved an accuracy of better than 70% at coverage levels of approximately 30% when using these learned information extractors to populate its university knowledge base while browsing new Web sites. These results provide encouraging initial support for our thesis, and suggest many routes for future research.

We have explored a variety of learning methods for this task, including statistical bag-of-words classifiers, first-order rule learners, and multi-strategy learning methods. We have found that statistical bag of words methods, derived from document classification methods in information retrieval, work well for classifying individual Web pages. However, these methods do not take advantage of the special hypertext structure available on the Web. Therefore, we developed first-order learning algorithms both for learning to classify pages and learning to recognize relations among several pages. These first-order methods are capable of describing patterns that occur across multiple Web pages, their hyperlinks, and specific words that appear on these pages and hyperlinks. Our experiments indicate that these methods tend to have higher accuracy than the bag of words classifiers, though they frequently provide lower coverage. In addition to these first-order learning methods that "look outward" from the page to consider its neighbors, we also have developed methods that "look inward" to consider the detailed structure of hypertext and specific text fragments within a single Web page. The SRV algorithm described in Section 7 learns relational rules that extract specific types of text fields within a Web page, such as a person's name.

We believe that the toolbox of methods we have described here will be applicable to a wide range of problem domains. For new domains, however, we may apply and combine the methods in ways not explored in this article. For example, in current work in a new problem domain, we are using page classifiers to recognize instances of a particular relation (the economic sector of a company), whereas in the work described here we used page classifiers to recognize class instances. In short, the most appropriate method for recognizing instances for a particular class or relation will depend on how these instances tend to be represented in the Web.

Based on the initial results reported here, we are optimistic about the future prospects for automatically constructing and maintaining a symbolic knowledge base by interpreting hypertext on the Web. Key questions remain, however. For example, what level of accuracy can be achieved by learned procedures for extracting information from the Web, and what level of accuracy will be required of them? For some tasks the required accuracy will be quite high (e.g., for an intelligent system that automatically invests money on behalf of its user). However, for tasks such as information retrieval on the Web, the system need only be sufficiently accurate to outperform the current keyword-based retrieval systems that have no real notion of an ontology. Although further research toward stronger learning methods is warranted, we conjecture that there will be a steady stream of applications where even an approximately correct knowledge base will outperform current keyword retrieval methods. A second type of question for our system is how much effort will be required to train the system for each new ontology, or for each extension to the growing ontology? In the experiments reported here, the system was trained using thousands of hand-labeled Web pages that were collected at a cost of approximately one or two person-weeks of effort. In newer work we are beginning to explore methods for reducing the dependence on hand labeled data. Below is a list of these and other research opportunities that merit further research:

- Develop learning methods that exploit the hierarchical relationships that exist among classes in the hierarchy. For example, in recent work we have shown that the accuracy of our Bayesian bag of words classifier can be improved by using the class hierarchy to obtain more accurate estimates of class conditional word probabilities [42].
- Use the vast pool of unlabeled Web pages to supplement the available hand-labeled data to improve learning accuracy. Recently we have shown that the EM algorithm can be used to combine labeled and unlabeled data to boost accuracy [48]. We are also exploring the combination of EM with pool-based training for *active learning* in which the learner requests labels for specific Web pages whose label will be especially helpful [41].
- Co-training multiple classifiers. For example, consider a problem setting in which one Web page classifier examines the words on the page, and a second classifier examines instead the words on the incoming hyperlinks to that page. In recent work, we have proposed a method by which each classifier acts as a trainer for the other, and we have provided initial experiments and theoretical analysis showing the promise of this approach [9].
- Exploit more linguistic structure. We plan to explore ways in which noun, verb, and prepositional phrases extracted from the text can be used as features for information extraction. We have conducted preliminary experiments that show improved accuracy in some cases when our bag of words representation is augmented by these extracted phrases [25]. We conjecture that such linguistic features will be even more useful for tasks with few words, such as classifying individual hyperlinks.
- Explore multiple strategies for learning to extract text fields from Web pages. We have developed a number of approaches to this task [20,21,23], including multi-strategy learning [22].

- Integrate statistical bag-of-words methods into first-order learning tasks. We have begun developing methods that augment first-order learning with the ability to use bag-of-words classifiers to invent new predicates for characterizing the pages and hyperlinks referenced in learned rules [60].
- Exploit more HTML structure. We plan to investigate the utility of representing the HTML structure of pages when learning rules for relation classification and information extraction. We have investigated one approach to representing HTML structure and exploiting it for learning tasks [16].
- Learn regularities over the growing knowledge base. We plan to use learning methods to discover interesting regularities over the facts that have been extracted from the Web, and to use these learned facts to improve future fact extraction. For example, in the university knowledge base we might expect to learn how to predict the department of a faculty member based on the department of her student advisees.
- Extend the ontology to new problem domains. We are currently applying our methods to the task of extracting information about companies from the Web.

A Assumptions About How Class and Relation Instances Are Represented

In this appendix we provide a formal treatment of the assumptions we make about how class and relation instances can be represented in the Web.

Let a Web page p be a sequence of tokens, $(t_1^p, t_2^p, ...t_n^p)$, where the tokens are words and HTML tags. Define the function $\mathsf{page}(t_s^p, ...t_e^p)$ so that it returns the page p in which a sequence of tokens resides.

Some tokens represent hyperlinks. Let the predicate $\mathsf{hyperlink}(t_i^p)$ be true if the ith token in page p is a hyperlink, and let the function $\mathsf{range}(t_i^p)$ return the page that is pointed to by a given hyperlink.

We define a relation $\mathsf{linked}(u, v)$ that represents the cases where there is a direct hyperlink from u to v, or where there is a path from u to v following the directed edges of hyperlinks:

(1) $\exists t_i^u$ such that $\mathsf{hyperlink}(t_i^u) = true$ and $\mathsf{range}(t_i^u) = v$, or
(2) $\exists w$ such that $\mathsf{linked}(u, w)$ and $\mathsf{linked}(w, v)$.

We define a relation $\mathsf{connected}(u, v)$ that represents the cases where there is a path from u to v following the *undirected* edges of hyperlinks:

(1) $\exists t_i^u$ such that $\mathsf{hyperlink}(t_i^u) = true$ and $\mathsf{range}(t_i^u) = v$, or
(2) $\exists t_i^v$ such that $\mathsf{hyperlink}(t_i^v) = true$ and $\mathsf{range}(t_i^v) = u$, or
(3) $\exists w$ such that $\mathsf{connected}(u, w)$ and $\mathsf{connected}(w, v)$.

Let us define a *segment* of hypertext as follows:

(1) A segment can consist of a subsequence of tokens in a page, $(t_s^p, ...t_e^p)$, where $1 \leq s \leq e \leq n$.
(2) A segment can consist of a page itself, $p = (t_1^p, t_2^p, ...t_n^p)$.
(3) A segment can consist of a rooted subgraph of pages: $P = p_0 \cup \{p_i \mid \exists p_j \in P,\ \mathsf{linked}(p_j, p_i)\}$. Here, p_0 represents the root of the subgraph.

We define a relation $\mathsf{contains}(u, v)$ that holds between two segments, u and v under the following conditions:

(1) both u and v are pages or sequences of tokens within pages, $t_s^u \geq t_s^v$, and $t_e^u \leq t_e^v$.
(2) u is a either a page or a sequence of tokens within a page, v is a rooted graph, and $\mathsf{page}(u) \in v$,
(3) both u and v are rooted graphs, and $u \subset v$.

Finally, let the predicate $\mathsf{model}_u(v)$ be true when the segment v satisfies an arbitrary statistical model representing v's relatedness to u.

Now, using these definitions, we state our assumptions about how class and relation instances can be represented on the Web.

- We assume that each instance of an ontology class is represented by one or more hypertext segments as defined above.
- We assume that each instance R(a,b) of a relation R is represented in one of three ways. Let s_a be the segment representing a and s_b be the segment representing b. Then R(a, b) can be represented by:

(1) $\mathsf{connected}(s_a, s_b)$,
(2) $\mathsf{contains}(s_a, s_b)$, or
(3) $\mathsf{model}_{\mathsf{b}}(s_a) = true$.

B Obtaining More Evenly Distributed Scores from Naive Bayes

While naive Bayes often provides accurate classifications, it presents problems when one wants to interpret the score for each class as an estimate of uncertainty. Per-class scores for the winning class tend to gravitate toward 1.0 and scores for the losing class tend toward 0.0. Often the effect is so strong that floating-point round-off error causes the probability to be calculated as exactly 1.0 for the winning class and 0.0 for the others. These extreme values are an artifact of the independence assumption. If for each word, the value of $\Pr(w|C)$ between different classes differs by one order of magnitude, then the final probabilities will differ by as many orders of magnitude as there are words in the document. Class-conditional word probabilities would be much more similar across classes if word dependencies were taken into account.

We would like scores that accurately reflect the uncertainty in each prediction and enable us to sensibly compare the scores of multiple documents. We attempt to counter the extreme values, while still avoiding the complexity of modeling word-dependencies, in two steps.

First, instead of using the product of the word likelihoods, we use the geometric mean of the likelihoods. This approach is closely related to the concept of *perplexity* in language modeling for speech recognition [54]. Perplexity is a measure of the likelihood of some data given a model, where the likelihood is normalized for the length of the data. We begin with naive Bayes (Eq. 5), rewrite the sum to an equivalent expression that sums over all words in the vocabulary T instead of just the words in the document (Eq. B.1), take the log, (Eq. B.2), and divide by the number of words in the document (Eq. B.3). This results in the log of the geometric mean of the word likelihoods, plus a term for the class prior.

$$\begin{aligned}
\Pr(c)\prod_{i=1}^{n}\Pr(w_i|c) &= \Pr(c)\prod_{i=1}^{T}\Pr(w_i|c)^{N(w_i,d)} && \text{(B.1)}\\
&\propto \log(\Pr(c)) + \sum_{i=1}^{T} N(w_i,d)\;\log(\Pr(w_i|c)) && \text{(B.2)}\\
&\propto \frac{\log(\Pr(c))}{n} + \sum_{i=1}^{T}\frac{N(w_i,d)}{n}\log(\Pr(w_i|c)) && \text{(B.3)}
\end{aligned}$$

If we interpret $N(w_i,d)/n$ as $\Pr(w_i|d)$, the right-hand term of this expression is the negative Cross Entropy [14] between the distribution of words induced by the document with the distribution of words induced by the class:

$$\frac{\log(\Pr(c))}{n} + \sum_{i=1}^{T} \Pr(w_i|d)\ \log(\Pr(w_i|c)). \tag{B.4}$$

Thus, the second term specifies that the class c with the highest score will be the one with the lowest Cross Entropy—the class that could "compress" the document most efficiently. This expression results in scores for each class that vary smoothly, without tendencies toward extreme values.

Cross Entropy in Equation B.4 can be intuitively understood as the average number of bits necessary to encode a word from the document using an encoding that is optimal for the distribution of words independently drawn from the class. Cross Entropy does not, however, account for the varying difficulty of encoding different documents—some documents are more complex, and inherently require more bits on average to encode. We want scores that can be sensibly compared between documents. A way to account for differences between documents is to use Kulback-Leibler Divergence—that is, to subtract the average number of bits it would take to encode the document using its optimal encoding (assuming again, that the words are independent of one another). This results in an expression that can be intuitively understood as the average number of *extra* bits required because we are using a suboptimal encoding instead of the optimal encoding. We modify the second term of Equation B.4 so that it expresses the KL Divergence score for each class:

$$\frac{\log(\Pr(c))}{n} + \sum_{i=1}^{T} \Pr(w_i|d)\ \log\left(\frac{\Pr(w_i|c)}{\Pr(w_i|d)}\right). \tag{B.5}$$

We also normalize the scores across all classes so that they sum to a constant. This normalization has the effect of increasing our confidence in the classification of documents with high word entropy. This is intuitively desirable because high-entropy documents have more unique words, which can be considered as stronger evidence, and more likely to result in correct classifications.

Note that the modifications to 5 do not change the ordering of class estimates for a given document. Consequently, the classifications made by naive Bayes are not affected. These modifications only serve to provide well-distributed, comparable scores.

Note that none of the changes since straightforward naive Bayes in Equation 5 has changed the scored ordering of different classes for the same document—they have not changed classification that would have resulted from naive Bayes. They have only served to provide well-distributed, comparable scores.

C The URL Grouping Algorithm

In this appendix, we present the details of the algorithm used to identify multi-page segments. As discussed in Section 5.4, we use the regularities in URL structure and naming to group related pages together, and to identify the any primary home pages in each group.

In the algorithm below, all wildcards must match some text. The wildcard * does not match across directory boundaries, but the wildcard @ does. The wildcard ? matches a single digit. The wildcard variable %1 does not match across directories.

- **Inputs:** A set of web pages, each with a URL, a tentative classification and a score.
- For each web page, identify its group:
  - The group is the longest prefix (indicated in parentheses) when the URL matches any of the patterns:
    - (@/{user,faculty,people,home,projects}/*)/*.{html,htm}
    - (@/{user,faculty,people,home,projects}/*)/
    - (@/{user,faculty,people,home,projects}/*)
    - (@/{cs???,www/,~*})/*.{html,htm}
    - (@/{cs???,www/,~*})/
    - (@/{cs???,www/,~*})
  - If no prefix matches, the group is the complete directory portion of the URL.
- For each group, identify at least one primary page:
  - A primary page is any page which URL matches:
    - @/index.{html,htm}
    - @/home.{html,htm}
    - @/homepage.{html,htm}
    - @/cs???.{html,htm}
    - @/%1/%1.{html,htm}
  - If no page in the group matches one of these patterns, then the page with the highest score for any non-`other` class is a primary page.
- Change the classification of all non-primary pages to `other`.
- **Outputs:** The final classification and score of each web page.

References

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

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

[3] *The Fourth Text Retrieval Cconference.* National Technical Information Services, Springfield, VA, 1995.

[4] *The Fifth Text Retrieval Cconference.* National Technical Information Services, Springfield, VA, 1996.

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

[6] C. Apté, F. Damerau, and S. M. Weiss. Towards language independent automated learning of text categorization models. Technical report, IBM, 1994. (TR 19481) (also appeared in SIGIR94).

[7] M. Balabanovic and Y. Shoham. Learning information retrieval agents: Experiments with automated web browsing. In *AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments*, 1995.

[8] E. Bloedorn, I. Mani, and T. R. MacMillan. Machine learning of user profiles: Representational issues. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 433–438. AAAI/MIT Press, 1996.

[9] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory.* ACM, 1998.

[10] T. Bray, J. Paoli, and C. M. Sperberg-McQueen. Extensible markup language (XML) 1.0. Technical Report REC-xml-19980210, World Wide Web Consortium, 1998. http://www.w3.org/TR/REC-xml.

[11] M. E. Califf and R. J. Mooney. Relational learning of pattern-match rules for information extraction. In *Working Papers of ACL-97 Workshop on Natural Language Learning*, 1997.

[12] B. Cestnik. Estimating probabilities: A crucial task in machine learning. In L. Aiello, editor, *Proceedings of the Ninth European Conference on Artificial Intelligence (ECAI-90)*, pages 147–150, Stockholm, Sweden, 1990. Pitman.

[13] W. W. Cohen. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning.* Morgan Kaufmann, 1995.

[14] T. H. Cover and J. A. Thomas. *Elements of Information Theory.* John Wiley and Sons, Inc., 1991.

[15] H.C.M. de Kroon, T.M. Mitchell, and E.J.H Kerckhoffs. Improving learning accuracy in information filtering. In *International Conference on Machine Learning - Workshop on Machine Learning Meets HCI (ICML-96)*, 1996.

[16] D. DiPasquo. Using HTML formatting to aid in natural language processing on the World Wide Web, June 1998. Senior thesis, Computer Science Department, Carnegie Mellon University (http://www.cs.cmu.edu/~WebKB/danthesis.ps.gz).

[17] P. Domingos and M. Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29:103–130, 1997.

[18] R. Doorenbos, O. Etzioni, and D. S. Weld. A scalable comparison-shopping agent for the world-wide web. In *Proceedings of the 1st International Conference on Autonomous Agents*. ACM, 1997.

[19] S. Džeroski and I. Bratko. Handling noise in inductive logic programming. In S.H. Muggleton and K. Furukawa, editors, *Proceedings of the Second International Workshop on Inductive Logic Programming (ILP-92)*, number TM-1182 in ICOT Technical Memorandum, pages 109–125, Tokyo, Japan, 1992. Institute for New Generation Computer Technology.

[20] D. Freitag. Using grammatical inference to improve precision in information extraction. In *ICML-97 Workshop on Automation Induction, Grammatical Inference, and Language Acquisition*, Nashville, TN, 1997. Morgan Kaufmann.

[21] D. Freitag. Information extraction from HTML: Application of a general learning approach. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, Madison, WI, 1998. AAAI Press.

[22] D. Freitag. Multistrategy learning for information extraction. In *Proceedings of the 15th International Conference on Machine Learning*, pages 161–169. Morgan Kaufmann, 1998.

[23] D. Freitag. Toward general-purpose learning for information extraction. In *Proceedings of COLING/ACL-98*, 1998.

[24] Jerome H. Friedman. On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1(1):55–77, 1997.

[25] J. Fuernkranz, T. Mitchell, and E. Riloff. A case study in using linguistic phrases for text categorization of the www. In *Working Notes of the AAAI/ICML Workshop on Learning for Text Categorization*, 1998. http://www.cs.cmu.edu/~WebKB/aaai-ws-aslog.ps.gz.

[26] C.-N. Hsu and M.-T. Dung. Wrapping semistructured web pages with finite-state transducers. In *Proceedings of the Conference on Automatic Learning and Discovery (CONALD-98)*, 1998.

[27] T. Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 143–151, Nashville, TN, 1997. Morgan Kaufmann.

[28] T. Joachims, D. Freitag, and T. Mitchell. Webwatcher: A tour guide for the World Wide Web. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 770–775, Nagoya, Japan, 1997. Morgan Kaufmann.

[29] B. Katz. From sentence processing to information access on the World Wide Web. In *Proceedings of the AAAI Spring Symposium on Natural Language Processing for the World Wide Web*, 1997.

[30] J.-T. Kim and D. I. Moldovan. Acquisition of linguistic patterns for knowledge-based information extraction. *IEEE Transactions on Knowledge and Data Engineering*, 1995.

[31] D. Koller and M. Sahami. Toward optimal feature selection. In *Proceedings of Thirteenth International Conference on Machine Learning*. Morgan Kaufmann, 1996.

[32] D. Koller and M. Sahami. Hierarchically classifying documents using very few words. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 170–178, Nashville, TN, 1997. Morgan Kaufmann.

[33] N. Kushmerick. *Wrapper Induction for Information Extraction*. PhD thesis, University of Washington, 1997. Tech Report UW-CSE-97-11-04.

[34] K. Lang. NewsWeeder: Learning to filter Netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*. Morgan Kaufmann, 1995.

[35] D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Inforamtion Retrieval (SIGIR94)*, 1994.

[36] D. Lewis, R. E. Shapire, J. P. Callan, and R. Papka. Training algorithms for linear text classifiers. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 298–306, 1996.

[37] D. D. Lewis and M. Ringuette. A comparison of two learning algorithms for text categorization. In *Third Annual Symposium on Document Analysis and Information Retrieval*, pages 81–93, 1994.

[38] H. Lieberman. Letizia: An agent that assists Web browsing. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 924–929. Morgan Kaufmann, 1995.

[39] S. Luke, L. Spector, D. Rager, and J. Hendler. Ontology-based Web agents. In W. L. Johnson, editor, *Proceedings of the 1st International Conference on Autonomous Agents*, pages 59–66. ACM, 1997.

[40] B. Masand, G. Linoff, and D. Waltz. Classifying news stories using memory based reasoning. In *Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR92)*, pages 59–65, 1992.

[41] A. McCallum and K. Nigam. Employing EM in pool-based active learning for text classification. In *Proceedings of the 15th International Conference on Machine Learning*, pages 350–358. Morgan Kaufmann, 1998.

[42] A. McCallum, R. Rosenfeld, T. Mitchell, and A. Ng. Improving text clasification by shrinkage in a hierarchy of classes. In *Proceedings of the 15th International Conference on Machine Learning*, pages 359–367. Morgan Kaufmann, 1998.

[43] A. K. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *Working Notes of the ICML/AAAI Workshop on Learning for Text Categorization*, 1998. http://www.cs.cmu.edu/∼mccallum.

[44] A. E. Monge and C. P. Elkan. The WEBFIND tool for finding scientific papers over the Worldwide Web. In *Proceedings of the Third International Congress on Computer Science Research*, Tijuana, Mexico, 1996.

[45] I. Moulinier and J.-G. Ganascia. Applying an existing machine learning algorithm to text categorization. In S. Wermter, E. Riloff, and G. Scheler, editors, *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing.* Springer-Verlag, March 1996.

[46] I. Moulinier, G. Raškinis, and J.-G. Ganascia. Text categorization: a symbolic approach. In *SDAIR*, 1996.

[47] Ion Muslea, Steve Minton, and Craig Knoblock. Wrapper induction for semistructured, Web-based information sources. In *Proceedings of the Conference on Automatic Learning and Discovery (CONALD-98)*, 1998.

[48] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 1999. To appear.

[49] M. J. Pazzani, J. Muramatsu, and D. Billsus. Syskill & Webert: Identifying interesting Web sites. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 54–59, Portland, OR, 1996. AAAI/MIT Press.

[50] M. Perkowitz and O. Etzioni. Category translation: learning to understand information on the Internet. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, pages 930–936. Morgan Kaufmann, 1995.

[51] P. Pirolli, J. Pitkow, and R. Rao. Silk from a sow's ear: Extracting usable structures from the Web. In *Human Factors in Computing Systems: CHI '96 Conference Proceedings*, pages 118–125, New York, NY, 1996.

[52] J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–2666, 1990.

[53] J. R. Quinlan and R. M. Cameron-Jones. FOIL: A midterm report. In *Proceedings of the European Conference on Machine Learning*, pages 3–20, Vienna, Austria, 1993.

[54] L. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition.* Prentice Hall Signal Processing Series, 1993.

[55] B. L. Richards and R. J. Mooney. Learning relations by pathfinding. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 50–55, San Jose, CA, 1992. AAAI/MIT Press.

[56] E. Riloff. Automatically generating extraction patterns from untagged text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1044–1049. AAAI/MIT Press, 1996.

[57] E. Riloff. An empirical study of automated dictionary construction for information extraction in three domains. *Arificial Intelligence*, 85:101–134, 1996.

[58] H. Schütze, D. A. Hull, and J. O. Pedersen. A comparison of classifiers and document representations for the routing problem. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR95)*, pages 229–237, 1995.

[59] M. Shakes, J. Langheinrich and O. Etzioni. Dynamic reference sifting: a case study in the homepage domain. In *Proceedings of Sixth International World Wide Web Conference*, Santa Clara, CA, 1996.

[60] S. Slattery and M. Craven. Combining statistical and relational methods for learning in hypertext domains. In *Proceedings of the 8th International Conference on Inductive Logic Programming.* Springer Verlag, 1998.

[61] S. Soderland. *Learning Text Analysis Rules for Domain-specific Natural Language Processing.* PhD thesis, University of Massachusetts, 1996. Available as Department of Computer Science Technical Report 96-087.

[62] S. Soderland. Learning to extract text-based information from the World Wide Web. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, 1997.

[63] Stephen Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1), February 1999.

[64] E. Spertus. ParaSite: Mining structural information on the Web. In *Proceedings of the Sixth International World Wide Web Conference*, Santa Clara, CA, 1997.

[65] S. M. Weiss and N. Indurkhya. Optimized rule induction. *IEEE Expert*, pages 61–69, December 1993.

[66] I. H. Witten and T. C. Bell. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4), July 1991.

[67] Y. Yang and J. Pederson. Feature selection in statistical learning of text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 412–420, Nashville, TN, 1997. Morgan Kaufmann.