Appears in *Proceedings of 15th National Conference on Artificial Intelligence (AAAI-98).*

Learning to Extract Symbolic Knowledge from the World Wide Web

Mark Craven[†] **Dan DiPasquo**[†] **Dayne Freitag**[†] **Andrew McCallum**[‡†]

Tom Mitchell[†] **Kamal Nigam**[†] **Seán Slattery**[†]

[†]School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213–3891
⟨firstname⟩.⟨lastname⟩@cs.cmu.edu

[‡]Just Research
4616 Henry Street
Pittsburgh, PA 15213

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* world wide 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: an ontology defining the classes and relations of interest, and a set of training data consisting of labeled regions of hypertext representing instances of these classes and relations. Given these inputs, the system learns to extract information from other pages and hyperlinks on the Web. This paper describes our general approach, several machine learning algorithms for this task, and promising initial results with a prototype system.

Introduction

The rise of the World Wide Web has made it possible for your workstation to retrieve any of 200 million Web pages for your personal perusal. The research described here is motivated by a simple observation: although your workstation can currently *retrieve* 200 million Web pages, it currently *understands* none of these Web pages. Of course this is because Web pages are written for human consumption and consist largely of text, images, and sounds. In this paper we describe a research effort with the long term goal of automatically creating and maintaining 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, and it would have many uses. At a minimum, it would allow much more effective information retrieval by supporting more sophisticated queries than current keyword-based search engines. Going a step further, it would enable new uses of the Web to support knowledge-based inference and problem solving.

How can we develop such a world wide knowledge base? The approach explored in our research is to develop a *trainable* system that can be taught to extract various types of information by automatically browsing the Web. This system accepts two types of inputs:

1. An ontology specifying the classes and relations of interest. 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).

2. Training examples that represent 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, our system attempts to learn general procedures for extracting new instances of these classes and relations from the Web.

To pursue this problem, we must make certain assumptions about the mapping between the ontology and the Web.

- We assume that each instance of an ontology class is represented by one or more contiguous segments of hypertext on the Web. By "contiguous segment of hypertext" we mean either a single Web page, or a contiguous string of text within a Web page, or a collection of several Web pages interconnected by 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 in an arbitrary Web page, or by a collection of interconnected Web pages that jointly describe the person.

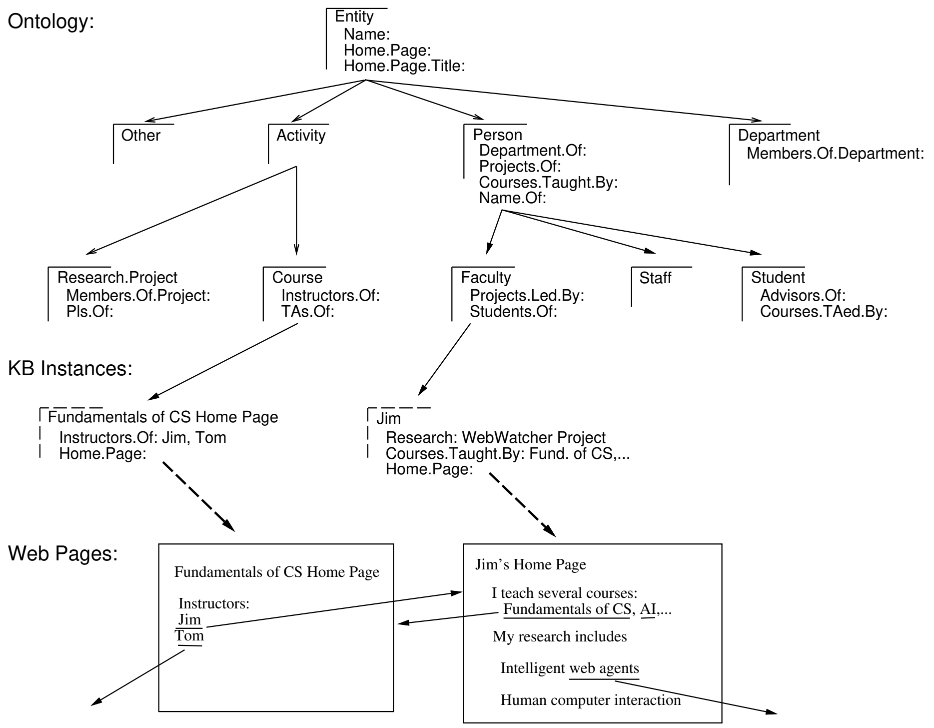

Figure 1: The inputs and outputs of the WebKB system. The top part of the figure shows an ontology that defines the classes and relations of interest. The bottom part shows two Web pages identified as training examples of the classes Course and Faculty. Together,these two pages also constitute a training example for the relations Instructors.Of and Courses.Taught.By. Given the ontology and a set of training data, WebKB 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.

- We assume that each instance R(A,B) of a relation R is represented on the Web in one of three ways. First, the instance R(A,B) may be represented by a segment of hypertext that *connects* the segment representing A to the segment representing B. For example, the hypertext segment shown at the bottom of Figure 1 connects the segment representing Jim with the segment representing Fundamentals.of.CS, and it represents the relation Instructor.Of.Course(Fundamentals.of.CS, Jim). Second, the instance R(A,B) may alternatively be represented by a contiguous segment of text representing A that *contains* the segment that represents B. For example, the relation instance Research.Of(Jim, Human.Computer.Interaction) is represented in Figure 1 by the fact that Jim's home page contains the phrase "Human computer interaction" in a specific 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.Of(Jim,Artificial.Intelligence) by classifying Jim's page using a statistical model of the words typically found in pages describing AI research.

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 paper. We plan to lift the following assumptions in the future as our research progresses.

- We assume 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.
- We assume that each class instance is represented by a *single* contiguous segment of hypertext. In other words, if the system encounters two non-contiguous Web pages that represent instances of the same class, it creates two distinct instances of this class in its knowledge base.

Given this problem definition and our current set of assumptions, we view the following as the three primary learning tasks involved in extracting knowledge-base instances from the Web: (i) recognizing class instances by classifying bodies of hypertext, (ii) recognizing relation instances by classifying chains of hyperlinks, (iii) recognizing class and relation instances by extracting small fields of text from Web pages. We discuss each of these tasks in the main sections of the paper. Additional details concerning the methods and experiments described in this paper can be found elsewhere (Craven *et al.* 1998). After describing approaches to these three tasks, we describe experiments with a system that incorporates learned classifiers for each task.

Experimental Testbed

As a testbed for our initial research, we have investigated the task of building a knowledge base describing computer science departments. As shown in Figure 1, our working ontology for this domain includes the classes Department, Faculty, Staff, Student, Research.Project, Course, and Other. Each of the classes has a set of slots defining relations that exist among instances of the given class and other class instances in the ontology.

We have assembled two data sets for the experiments reported here. The first is a set of pages and hyperlinks drawn from four CS departments. 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.

In addition to labeling pages, we also hand-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.Course relation consists of a Course home page and a Person home page. Our data set of relation instances comprises 251 Instructors.Of.Course instances, 392 Members.Of.Project instances, and 748 Department.Of.Person instances.

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. These heuristics were conservative, and thus we believe that, although some name occurrences were missed, there were no false positives. From a set of 174 Person pages, this procedure yielded 525 distinct name occurrences.

Recognizing 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 approach to classifying Web pages using the words found in pages. In the second part of this section we discuss learning first-order rules to classify Web pages. Finally, we consider using information from URLs to improve our page classification accuracy.

Statistical Text Classification

Our statistical page-classification method 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.

As is common in learning text classifiers, the probabilistic models we use ignore the sequence in which the words occur. These models are often called *unigram* or *bag-of-words* models because they are based on statistics about single words in isolation.

The approach that we use for classifying Web pages is the *naive Bayes* method, with minor modifications based on Kullback-Leibler Divergence. More precisely, we classify a document d as belonging to class c' according to the following rule:

$$c' = \underset{c}{\operatorname{argmax}} \left[\frac{\log \Pr(c)}{n} + \sum_{i=1}^{T} \Pr(w_i|d) \, \log \left(\frac{\Pr(w_i|c)}{\Pr(w_i|d)} \right) \right]$$

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 of drawing w_i given a document from class c, and $Pr(w_i|d)$ represents the frequency of occurrence of w_i in document d.

This method makes exactly the same classifications as naive Bayes, but produces classification scores that are less extreme, and thus better reflect uncertainty than those produced by naive Bayes.

When estimating the word probabilities, $\Pr(w_i|c)$, we use a smoothing method that prevents words from having zero probability and provides more robust estimates for infrequently occurring words. We have found that we get more accurate classifications when using a restricted vocabulary size, and thus we limit our vocabulary to 2000 words in our experiments. The vocabulary is selected by ranking words according to their average mutual information with respect to the class labels.

We evaluate our method using a four-fold cross-validation methodology. We conduct four runs in which we train classifiers using data from three of the universities in our data set (plus the auxiliary set of pages mentioned in the previous section), and test the classifiers using the university held out. On each iteration we hold out a different university for the test set.

Along with each classification, we calculate an associated measure of confidence which is simply the classification score described in the formula above. 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 10% of the pages available on the Web. The effect of trading off coverage for accuracy is shown in Figure 2. The horizontal axis on this plot represents *coverage*: the percentage of pages for a given class that are correctly classified as belonging to the class. The vertical axis represents *accuracy*:

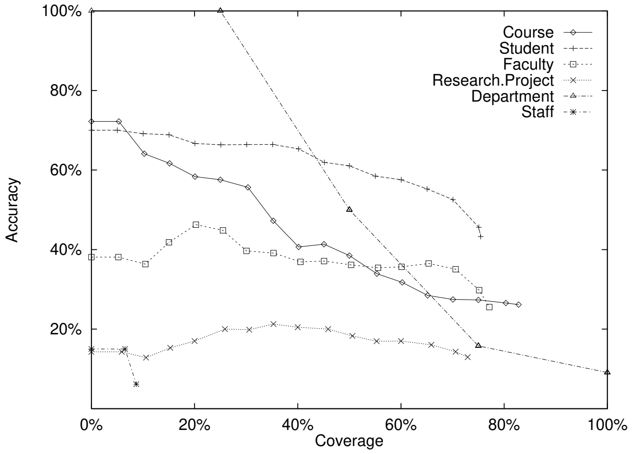

Figure 2: Accuracy/coverage for statistical classifiers.

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. If we accepted our classifiers' decisions every time they predicted Student, they would be correct 43% of the time. As we raise the confidence threshold for this class, however, the accuracy of our predictions rises. For example, at a coverage of 20%, accuracy reaches a level of 67%.

Nearly all of the misclassifications made by our statistical text classifiers involve two types of mistakes. First, the classifiers often confuse different subclasses of Person. For example, although only 9% of the Staff instances are correctly assigned to the Staff category, 80% of them are correctly classified into the more general class 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 if we correctly assign it to a more general class.

Second, the most common form of mistake involves classifying Other pages into one of the "core" classes; only 35% of Other instances are correctly classified. The low level of classification accuracy for the Other class is largely explained by the nature of the class. 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. We return to this issue shortly.

One of the interesting aspects of Web page classification, in contrast to conventional flat-text classification, is that redundancy of hypertext naturally suggests a variety of different representations for page classification. In addition to classifying a page using the words that occur in the page, we have also investigated classification using (a) the words that occur in hyperlinks (i.e. the words in the anchor text) that point to the page, and (b) the words that occur only in the HTML title and headings fields of the page. For some classes, these methods provide more accurate predictions than the approach described above. Space limitations preclude us from discussing these results in detail. In the next section, however, we describe another approach to Web page classification that exploits the special properties of hypertext.

First-Order Text Classification

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 method for classifying Web pages discussed above considers the words in either a single node of the graph or in a set of edges impinging on the same node. However, such methods do not allow us to learn models that take into account 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, which are able to represent general characteristics of a graph, require a first-order representation. In this section, we consider the task of learning to classify pages using an algorithm that is able to induce first-order rules.

The learning algorithm that we use in this section is FOIL (Quinlan & Cameron-Jones 1993). FOIL is a greedy covering algorithm for learning function-free Horn clauses. The representation we provide to the learning algorithm consists of the following background relations:

- has_*word*(Page): Each of these Boolean predicates indicates the pages in which the word *word* occurs.
- link_to(Page, Page): This relation represents the hyperlinks that interconnect the pages in the data set. The first argument is the page on which the link occurs, and the second is the page to which it is linked.

We apply FOIL to learn a separate set of clauses for six of the seven classes considered in the previous section. 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. The confidence of a prediction is determined

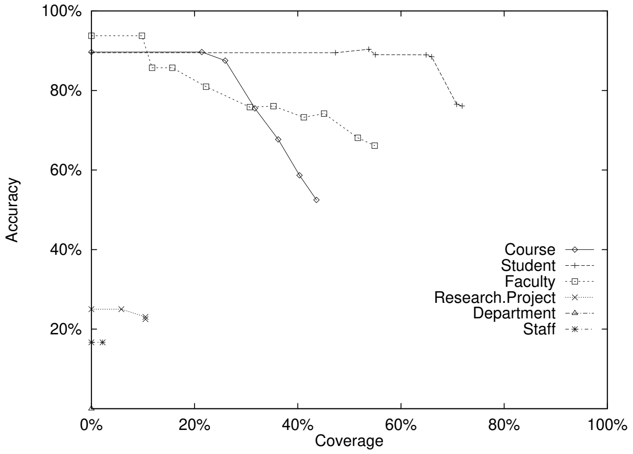

Figure 3: Accuracy/coverage for FOIL classifiers.

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

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

Figure 4: Two of the rules learned by FOIL for classifying pages, and their test-set accuracies.

by an m-estimate (Cestnik 1990) of the error-rate of the clause making the prediction. The resulting Accuracy/Coverage plot is shown in Figure 3. Comparing this figure to Figure 2, one can see that for several of the classes, the first-order rules are significantly more accurate than the statistical classifiers, although in general, their coverage is not quite as good.

The Student class provides an interesting illustration of the power of a first-order representation for learning to classify Web pages. Figure 4 shows the most accurate rule learned for this class for one of the training/test splits. 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) has learned to exploit "student directory" pages in order to identify student home pages. As this example shows, Web-page classification is different than ordinary text classification in that neighboring pages may provide strong evidence about the class of a page.

Identifying Multi-Page Segments

As discussed above, 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 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 classes) that were described by a 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 relax this simplifying assumption we must use methods for identifying sets of interlinked pages that represent a single knowledge base instance. Spertus (1997) has described regularities in URL structure and naming, and presented several heuristics for discovering page groupings and identifying representative home pages. We use a similar, slightly expanded, heuristic approach.

The impact of using the URL heuristics with our statistical text classifiers is summarized in Figure 5. Comparing these curves to Figure 2 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.

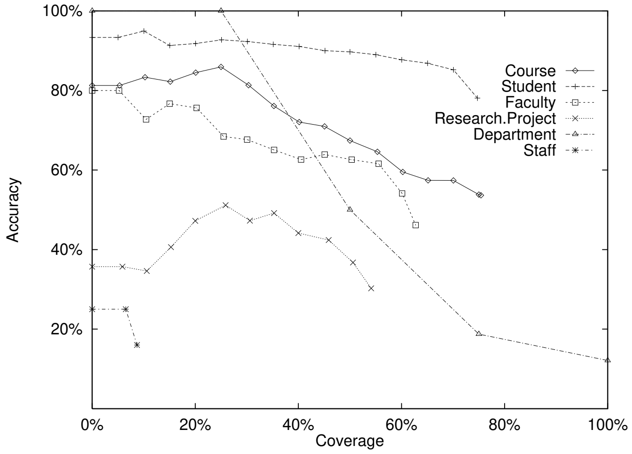

Figure 5: Accuracy/coverage for the statistical text classifiers after the application of URL heuristics.

Recognizing 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 relation instance involves inducing rules that characterize the prototypical paths of the relation.

```
members_of_project(A,B) :- research_project(A), person(B),
        link_to(C,A,D), link_to(E,D,B),
        neighborhood_word_people(C).
Test Set:  18 Pos, 0 Neg

department_of_person(A,B) :- person(A), department(B),
        link_to(C,D,A), link_to(E,F,D), link_to(G,B,F),
        neighborhood_word_graduate(E).
Test Set:  371 Pos, 4 Neg
```

Figure 6: Two of the rules learned for recognizing relation instances, and their test-set accuracies.

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. The representation consists of the following background relations:

- *class*(Page) : For each *class* from the previous section, the corresponding relation lists the pages that represent instances of *class*. These instances are determined using actual classes for pages in the training set and predicted classes for pages in the test set.
- link_to(Hyperlink, Page, Page) : This relation represents the hyperlinks that interconnect the pages in the data set.
- has_*word*(Hyperlink) : This set of relations indicates the words that are found in the anchor (i.e., underlined) text of each hyperlink.
- 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.
- 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.

We learn definitions for the members_of_project(Page, Page), instructors_of_course(Page, Page), and department_of_person(Page, Page) target relations. In addition to the positive instances, our training sets include approximately 300,000 negative examples.

The algorithm we use for learning to recognize relation instances is similar to FOIL. Unlike FOIL however, our method does not simply use a hill-climbing search when learning clauses. We have found that 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.

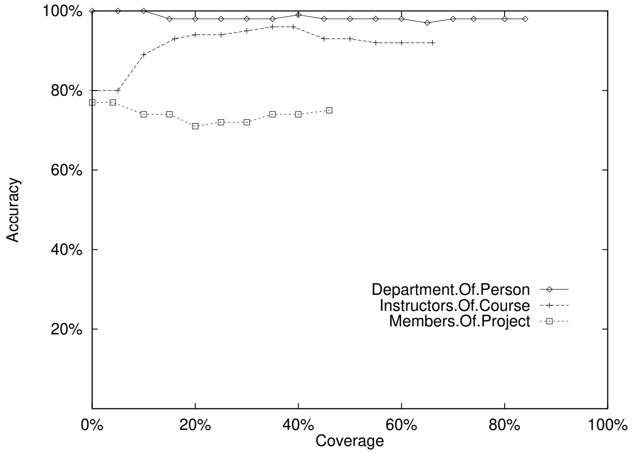

Figure 7: Accuracy/coverage for learned relation rules.

Our algorithm for constructing the path part of a clause is a variant of Richards and Mooney's (1992) *relational pathfinding* method.

Figure 6 shows one of the learned clauses for each of the Members.Of.Project and Department.Of.Person relations. 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 rule shown for the Members.Of.Project relation describes 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.Person 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. Using these confidence measures, Figure 7 shows the test-set accuracy/coverage curves for the three target relations. The accuracy levels of all three rule sets are fairly high. The limited coverage levels of the learned rules is due primarily to the limited coverage of our page classifiers since all of the learned rules include literals which test predicted page classifications.

Extracting Text Fields

In some cases, the information we want to extract will not be represented by Web pages or relations among pages, but instead it will be represented by small fragments of text embedded in pages. This type of task is commonly called *information extraction*. In this section we discuss our approach to learning rules for such information-extraction tasks.

We have developed an information-extraction learning algorithm called SRV which is a hill-climbing, first-order learner in the spirit of FOIL. Input to the algorithm is a set of pages labeled to identify instances of the field we want to extract. 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.

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 predicate using all such negative examples.

The representation used by our rule learner includes the following relations:

- length(Fragment, Relop, N): The learner can specify the length of a field, in terms of number of tokens, is less than, greater than, or equal to some integer.
- some(Fragment, Var, Path, Attr, Value): The learner can posit an attribute-value test for some token in the sequence (e.g., "the field contains some token that is capitalized"). One argument to this predicate is a variable. Each such variable binds to a distinct token. Thus, if the learner uses a variable already in use in the current rule, it is specializing the description of a single token; if the variable is a new one, it describes a previously unbound token. The learner has the option of adding an arbitrary path of relational attributes to the test, so that it can include literals of the form, "some token which is followed by a token which is followed by a token that is capitalized."
- position(Fragment, Var, From, Relop, N): The learner can say something about the position of a token bound by a *some*-predicate in the current rule. The position is specified relative to the beginning or end of the sequence.
- relpos(Fragment, Var1, Var2, Relop, N): Where at least two variables have been introduced by *some*-predicates in the current rule, the learner can specify their ordering and distance from each other.

As in the previous experiments, we follow the leave-one-university-out methodology. 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.

Figure 8 shows a learned rule and its application to a test case. Figure 9 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' 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%.

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

```
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>
```

Figure 8: **Top:** An extraction rule for name of home page owner. This rule looks for 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, unknown from the training data, and not a four-character token. **Bottom:** An example HTML fragment which the above rule matches.

The Crawler

The previous sections have considered the tasks of learning to recognize class and relation instances in an off-line setting. In this section, we describe an experiment that involves evaluating our approach in a novel, on-line environment.

We have developed a Web-crawling system that populates a knowledge base with class and relation instances as it explores the Web. The system incorporates trained classifiers for the three learning tasks discussed previously: recognizing class instances, recognizing relation instances, and extracting text fields. Our crawler employs a straightforward strategy to browse the Web. It maintains a priority queue of pages to be explored. Each time it processes a Web page, it considers adding the URLs of the hyperlinks found on this page to the queue. One of these URLs is added if (1) the current page led to the creation of a new instance in the knowledge base, and (2) the URL is within the domain allowed by a user-specified parameter.

To evaluate the performance of our crawler, we trained a set of classifiers using all of the data in our four-university set and the auxiliary set, and then gave the system the task of exploring a fifth Web site: the computer science department at Carnegie Mellon Uni-

	Student	Faculty	Person	Project	Course	Dept.	Instruct.Of	Members.Of.Project	Department.Of
Extracted	180	66	246	99	28	1	23	125	213
Correct	130	28	194	72	25	1	18	92	181
Accuracy	72%	42%	79%	73%	89%	100%	78%	74%	85%

Table 1: Page and relation classification accuracy when exploring the CMU computer science department Web site.

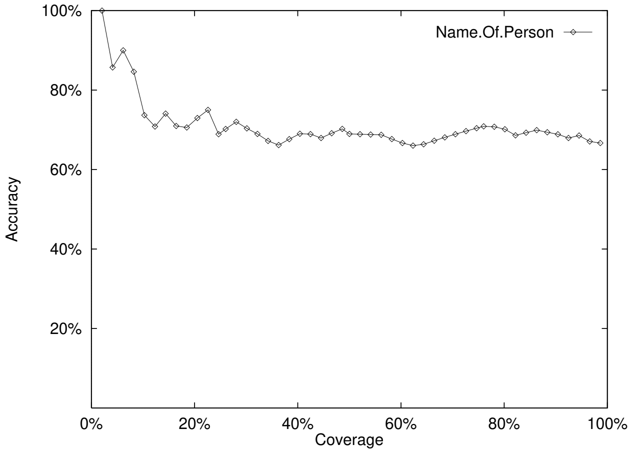

Figure 9: Accuracy/coverage for learned name-extraction rules.

versity. After exploring 2722 Web pages at this site, the crawler extracted 374 new class instances and 361 new relation instances for its knowledge base. The accuracy of the crawler over this run is summarized in Table 1. The name extractor produced a name for each of the 246 extracted Person instances. Among the 194 pages that actually represented people, 73% of the names were correctly identified. Overall its accuracy was 57%.

This experiment confirms that the learned classifiers described earlier in this paper can be used to accurately populate a knowledge base in an on-line setting.

Related Work

Our work builds on related research in several fields, including text classification (e.g. Lewis *et al.*, 1996), information extraction (e.g. Soderland, 1996), and Web agents (e.g. Shakes & Etzioni, 1996). Space limitations preclude us from describing this work in detail; the interested reader is referred elsewhere (Craven *et al.* 1998) for a comprehensive discussion of related work.

Conclusions

We began with the question of how to automatically create a computer-understandable world-wide knowledge base whose content mirrors that of the World Wide Web. The approach we propose in this paper is to construct a system that can be trained to automatically populate such a knowledge base.

The key technical problem in our proposed approach is to develop accurate learning methods for this task. We have presented a variety of approaches that take advantage of the special structure of hypertext by considering relationships among Web pages, their hyperlinks, and specific words on individual pages and hyperlinks.

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. Currently, we are extending our system to handle a richer ontology, and we are investigating numerous research issues such as how to reduce training data requirements, how to exploit more linguistic and HTML structure, and how to integrate statistical and first-order learning techniques.

Acknowledgments

This research is supported in part by the DARPA HPKB program under contract F30602-97-1-0215.

References

Cestnik, B. 1990. Estimating probabilities: A crucial task in machine learning. In Aiello, L., ed., *Proc. of the 9th European Conf. on Artificial Intelligence.*

Craven, M.; DiPasquo, D.; Freitag, D.; McCallum, A.; Mitchell, T.; Nigam, K.; and Slattery, S. 1998. Learning to extract symbolic knowledge from the World Wide Web. Technical report, CMU CS Dept.

Lewis, D.; Schapire, R. E.; Callan, J. P.; and Papka, R. 1996. Training algorithms for linear text classifiers. In *Proc. of the 19th Annual Int. ACM SIGIR Conf.*

Quinlan, J. R., and Cameron-Jones, R. M. 1993. FOIL: A midterm report. In *Proc. of the 12th European Conf. on Machine Learning.*

Richards, B. L., and Mooney, R. J. 1992. Learning relations by pathfinding. In *Proc. of the 10th National Conf. on Artificial Intelligence.*

Shakes, J. Langheinrich, M., and Etzioni, O. 1996. Dynamic reference sifting: a case study in the homepage domain. In *Proc. of 6th Int. World Wide Web Conf.*

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

Spertus, E. 1997. ParaSite: Mining structural information on the Web. In *Proc. of the 6th Int. World Wide Web Conf.*