, , 1–25 ()

Relational Learning with Statistical Predicate Invention: Better Models for Hypertext

MARK CRAVEN craven@biostat.wisc.edu
Department of Biostatistics & Medical Informatics, University of Wisconsin, Madison, WI 53706

SEÁN SLATTERY jslttery@cs.cmu.edu
School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213

Editor:

Abstract. We present a new approach to learning hypertext classifiers that combines a statistical text-learning method with a relational rule learner. This approach is well suited to learning in hypertext domains because its statistical component allows it to characterize text in terms of word frequencies, whereas its relational component is able to describe how neighboring documents are related to each other by hyperlinks that connect them. We evaluate our approach by applying it to tasks that involve learning definitions for (i) classes of pages, (ii) particular relations that exist between pairs of pages, and (iii) locating a particular class of information in the internal structure of pages. Our experiments demonstrate that this new approach is able to learn more accurate classifiers than either of its constituent methods alone.

Keywords: Relational Learning, Text Categorization, Predicate Invention, Naive Bayes

1. Introduction

In recent years there has been a great deal of interest in applying machine-learning methods to a variety of problems in classifying and extracting information from text. In large part, this trend has been sparked by the explosive growth of the World Wide Web. An interesting aspect of the Web is that it can be thought of as a graph in which pages are the nodes of the graph and hyperlinks are the edges. The graph structure of the Web makes it an interesting domain for relational learning. In previous work (Craven, Slattery, & Nigam, 1998b), we demonstrated that for several Web-based learning tasks, a relational learning algorithm can learn more accurate classifiers than a common statistical approach. In this paper, we present a new approach to learning hypertext classifiers that combines a statistical text-learning method with a relational rule learner. We present experiments that evaluate one particular instantiation of this general approach: a FOIL-based (Quinlan, 1990; Quinlan & Cameron-Jones, 1993) learner augmented with the ability to invent predicates using a Naive Bayes text classifier. Our experiments indicate that this approach is able to learn classifiers that are often more accurate than either purely statistical or purely relational alternatives.

In previous research, the Web has provided a fertile domain for a variety of machine-learning tasks, including learning to assist users in searches (Joachims, Freitag, & Mitchell, 1997), learning information extractors (Kushmerick, Weld, & Doorenbos, 1997; Soderland, 1997), learning user interests (Mladenić, 1996; Paz-

zani, Muramatsu, & Billsus, 1996), and others. Most of the research in this field has involved (i) using propositional or statistical learners, and (ii) representing documents by the words that occur in them. Our approach is motivated by two key properties of hypertext:

- Documents (i.e. pages) are related to one another by hyperlinks. Important sources of evidence for Web learning tasks can often be found in neighboring pages and hyperlinks.
- Large feature sets are needed to represent the content of documents because natural language involves large vocabularies. Typically, text classifiers have feature spaces of hundreds or thousands of words.

Because it uses a relational learner, our approach is able to represent document relationships (i.e. arbitrary parts of the hypertext graph) in its learned definitions. Because it also uses a statistical learner with a feature-selection method, it is able to learn accurate definitions in domains with large vocabularies. Although our algorithm was designed with hypertext in mind, we believe it is applicable to other domains that involve both relational structure and large feature sets.

In the next section we describe the commonly used *set-of-words* and *bag-of-words* representations for learning text classifiers. We describe the use of bag-of-words with the Naive Bayes algorithm, which is often applied to text learning problems. We then describe how a relational learner, such as FOIL, can use a set-of-words representation along with background relations describing the connectivity of pages for hypertext learning tasks. In Section 3, we describe our new approach to learning in hypertext domains. Our method is based on the Naive Bayes and FOIL algorithms presented in Section 2. In Section 4 we empirically evaluate our algorithm on three types of tasks – learning definitions of page classes, learning definitions of relations between pages, and learning to locate a particular type of information within pages – that we have investigated as part of an effort aimed at developing methods for automatically constructing knowledge bases by extracting information from the Web (Craven, DiPasquo, Freitag, McCallum, Mitchell, Nigam, & Slattery, 1998a). Finally, Section 6 provides conclusions and discusses future work.

2. Two Approaches to Hypertext Learning

In this section we describe two approaches to learning in text domains. First we discuss the *Naive Bayes* algorithm, which is commonly used for text classification, and then we describe an approach that involves using a relational learning method, such as FOIL(Quinlan, 1990; Quinlan & Cameron-Jones, 1993), for such tasks. These two algorithms are the constituents of the hybrid algorithm that we present in section 3.

2.1. Naive Bayes for Text Classification

Most work in learning text classifiers involves representing documents as either *sets of words* or *bags of words*. Both of these are based on a vector representation of

documents, with each element corresponding to a distinct word. The set-of-words representation indicates only word presence or absence in the document, while the bag-of-words representation takes the frequency of the word in the document into account. The key assumption made by these representations is that the position of a word in a document does not matter (i.e. encountering the word *machine* at the beginning of a document is the same as encountering it at the end).

The Naive Bayes classifier with a bag-of-words document representation is commonly used for text classification (Mitchell, 1997). Given a document d with n words $(w_1, w_2, \ldots, w_n)$, we can determine the probability that d belongs to the jth class, c_j, as follows:

$$\Pr(c_j|d) = \frac{\Pr(c_j)\Pr(d|c_j)}{\Pr(d)} \approx \frac{\Pr(c_j)\prod_{i=1}^{n}\Pr(w_i|c_j)}{\Pr(d)}. \quad (1)$$

Using this method to classify a document into one of a set of classes C, we simply calculate:

$$\arg\max_{c_j \in C} \Pr(c_j)\prod_{i=1}^{n}\Pr(w_i|c_j). \quad (2)$$

In order to make the word probability estimates $\Pr(w_i|c_j)$ robust with respect to infrequently encountered words, it is common to use a smoothing method to calculate them. Once such smoothing technique is to use Laplace estimates:

$$\Pr(w_i|c_j) = \frac{N(w_i, c_j) + 1}{N(c_j) + T}$$

where $N(w_i, c_j)$ is the number of times word w_i appears in training set examples from class c_j, $N(c_j)$ is the total number of words in the training set for class c_j and T is the total number of unique words in the corpus.

In addition to the position-independence assumption implicit in the bag-of-words representation, Naive Bayes makes the additional assumption that the occurrence of a given word in a document is independent of all other words in the document, given the class. Clearly, this assumption does not hold in real text documents. However, in practice Naive Bayes classifiers often perform quite well (Lewis & Ringuette, 1994).

Since document corpora typically have vocabularies of thousands of words, it is common in text learning to use some type of feature selection method. Frequently used methods include

(i) dropping putatively un-informative words that occur on a *stop-list*,

(ii) dropping words that occur fewer than a specified number of times in the training set,

(iii) ranking words by a measure such as their mutual information with the class variable, and then dropping low-ranked words (Yang & Pedersen, 1997), and

(iv) *stemming.* Stemming refers to the process of heuristically reducing words to their root form. For example, using the Porter stemmer (Porter, 1980), the words *compute*, *computers* and *computing* would be stemmed to the root *comput*. Even after employing such feature-selection methods, it is common to use feature sets consisting of hundreds or thousands of words.

Note that while direct application of Formula 2 provides a classification for a document, we can use the probabilities generated as a measure of confidence in a predicted class. Depending on the intended application of our learned models, we may want to accept only some of our most confident predictions. In the experiments presented in Section 4, we consider how the predictive accuracy of various learned models changes as we limit our predictions by thresholding on confidence.

2.2. Relational Text Learning

Both propositional and relational symbolic rule learners have also been used for text learning tasks (Cohen, 1995a, 1995b; Moulinier, Raškinis, & Ganascia, 1996). We argue that relational learners are especially appealing for learning in hypertext domains because they enable learned classifiers to represent the relationships among documents as well as information about the occurrence of words in documents. In previous work (Craven et al., 1998b), we demonstrated that this ability enables relational methods to learn more accurate classifiers than propositional methods in some cases.

In Section 4, we present experiments in which we apply FOIL to several hypertext learning tasks. The problem representation we use for our relational learning tasks consists of the following background relations:

- link_to(Hyperlink, Page, Page, Tag) : This relation represents Web hyperlinks. For a given hyperlink, the first argument 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. The fourth argument encodes whether the link points to a page on another site (OFFSITE), a page in a subdirectory (DOWN), a page in the current directory (LATERAL), a page in a parent directory (UP) or a page in a subdirectory of a parent directory (UPDOWN).

- has_*word*(Page) : This set of relations indicates the words that occur on each page. There is one predicate for each word in the vocabulary, and each instance indicates an occurrence of the word in the specified page.

- has_anchor_*word*(Hyperlink) : This set of relations indicates the words that are found in the anchor (i.e., underlined) text of each hyperlink.

- 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.

- all_words_capitalized(Hyperlink) : The instances of this relation are those hyperlinks in which all 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).

This representation for hypertext enables the learner to construct definitions that describe the graph structure of the Web (using the link_to relation) and word occurrences in pages and hyperlinks. A set-of-words representation of pages and hyperlinks is provided by the has_*word*, has_anchor_*word* and has_neighborhood_*word* predicates. Note that we do not use theory constants to represent words because doing so would require the relational learner we use (FOIL) to take two search steps in order to add a word-test literal. In our representation, such a test can be added in a single step.

Unlike Naive Bayes, FOIL does not provide a standard method for ordering its predictions by confidence. In section 4.3 we will show one scheme for producing prediction confidences with this learner.

3. Combining the Statistical and Relational Approaches

In this section we present an approach that combines a statistical text learner with a relational learner. We argue that this algorithm is well suited to hypertext learning tasks. Like a conventional bag-of-words text classifier, our algorithm is able to learn predicates that characterize pages or hyperlinks by their word statistics. However, because it is a relational learning method, it is also able to represent the graph structure of the Web, and thus it can represent the word statistics of neighboring pages and hyperlinks.

As described in the previous section, a conventional relational learning algorithm, such as FOIL, can use a set-of-words representation when learning in hypertext domains. We hypothesize, however, that our algorithm has two properties that make it better suited to such tasks than an ordinary relational method:

- Because it characterizes pages and hyperlinks using a statistical method, its learned rules will not be as dependent on the presence or absence of specific key words as a conventional relational method. Instead, the statistical classifiers in its learned rules consider the weighted evidence of many words.

- Because it learns each of its statistical predicates to characterize a specific set of pages or hyperlinks, it can perform feature selection in a more directed manner. The vocabulary to be used when learning a given predicate can be selected specifically for the particular classification task at hand. In contrast, when selecting a vocabulary for a relational learner that represents words using background relations, the vocabulary is pruned without regard to the particular subsets of pages and hyperlinks that will be described in clauses, since *a priori* we do not know which subsets it will be useful for the learner to describe.

Input: uncovered positive examples T^+ of target relation R,
all negative examples T^- of target relation R,
background relations

1. initialize clause C: $R(X_0, ...X_k)$:- *true*.
2. $T = T^+ \cup T^-$
3. while T contains negative tuples and C is not too complex
4. call predicate-invention method to get new candidate literals (Table 2)
5. select literal (from background or invented predicates) to add to C
6. update tuple set T to represent variable bindings of updated C
7. for each invented predicate $P_j(X_i)$
8. if $P_j(X_i)$ was selected for C then retain it as a background relation

Return: learned clause C

Table 1. The inner loop of FOIL-PILFS. This is essentially the inner loop of FOIL augmented with our predicate invention procedure.

We consider our approach to be quite general: it involves using a relational learner to represent graph structure, and a statistical learner with a feature-selection method to characterize the edges and nodes of the graph. Here we present an algorithm, which we refer to as FOIL-PILFS (for FOIL with Predicate Invention for Large Feature Spaces), that represents one particular instantiation of our approach. This algorithm is basically FOIL, augmented with a predicate-invention method in the spirit of CHAMP (Kijsirikul, Numao, & Shimura, 1992). Table 1 shows the inner loop of FOIL-PILFS (which learns a single clause) and its relation to its predicate invention method, which is shown in Table 2. Aside from the steps numbered 4, 7, and 8, the inner loop of FOIL-PILFS is the same as the inner loop of FOIL.

The predicates that FOIL-PILFS invents are statistical classifiers applied to some textual description of pages, hyperlinks, or components thereof. Currently, the invented predicates are only unary, boolean predicates. We assume that each constant in the problem domain has a type, and that each type may have one or more associated document collections. Each constant of the given type maps to a unique document in each associated collection. For example, the type *page* might be associated with a collection of documents that represent the words in pages, and the type *hyperlink* might be associated with two collections of documents – one which represents the words in the anchor text of hyperlinks and one which represents the "neighboring" words of hyperlinks.

Whereas CHAMP considers inventing a new predicate only when the basic relational algorithm fails to find a clause, our method considers inventing new predicates at each step of the search for a clause. Specifically, at some point in the search, given a partial clause C that includes variables X_1,..., X_n, our method considers inventing predicates to characterize each X_i for which the variable's type has an

Input: partial clause C,
document collection for each type,
parameter ϵ

1. for each variable X_i in C
2. for each document collection D_j associated with the *type* of X_i
3. S^+ = documents in D_j representing constants bound to X_i in pos tuples
4. S^- = documents in D_j representing constants bound to X_i in neg tuples
5. rank each word in $S^+ \cup S^-$ according to mut. info. w/ class variable
6. $n = |S^+ \cup S^-| \times \epsilon$
7. F = top ranked n words
8. call Naive Bayes to learn $P_j(X_i)$ w/ feature set F, training set $S^+ \cup S^-$

Return: all learned predicates $P_j(X_i)$

Table 2. The FOIL-PILFS predicate invention method.

associated collection of documents. If there is more than one document collection associated with a type, then we consider learning a predicate for each collection. For example, if X_i is of type *hyperlink*, and we have two document collections associated with *hyperlink* – one for anchor text and one for "neighboring" text – then we would consider learning one predicate to characterize the constants bound to X_i using their anchor text, and one predicate to characterize the constants using their "neighboring" text.

Once the method has decided to construct a predicate on a given variable X_i using a given document collection, the next step is to assemble the training set for the Naive Bayes learner. If we think of the tuple set currently covered by C as a table in which each row is a tuple and each column corresponds to a variable in the clause, then the training set consists of those constants appearing in the column associated with X_i. Each row corresponds to either the extension of a positive training example or the extension of a negative example. Thus those constants that appear in positive tuples become positive instances for the predicate-learning task and those that appear in negative tuples become negative instances.

One issue that crops up, however, is that a given constant might appear multiple times in the X_i column, and it might appear in both positive and negative tuples. We enforce a constraint that a constant may appear only once in the predicate's training set. For example, if a given constant is bound to X_i in multiple positive tuples, it appears as only a single instance in the training set for a predicate. The motivation for this choice is that we want to learn Naive Bayes classifiers that generalize well to new documents. Thus we want the learner to focus on the characteristics that are common to many of the documents in the training set, instead of focusing on the characteristics of a few instances that each occur many times in the training set.

Before learning a predicate using this training set, our method determines the vocabulary to be used by Naive Bayes. In some cases the predicate's training set may consist of a small number of documents, each of which might be quite large. Thus, we do not necessarily want to allow Naive Bayes to use all of the words that occur in the training set as features. The method that we use involves the following two steps. First, we rank each word w_i that occurs in the predicate's training set according to its mutual information with the target class for the predicate. Second, given this ranking, we take the vocabulary for the Naive Bayes classifier to be the n top-ranked words where n is determined as follows:

$$n = \epsilon \times m \tag{3}$$

Here m is the number of instances in the predicate's training set, and ϵ is a parameter (set to 0.1 throughout our experiments).

The motivation for this heuristic is the following. We want to make the dimensionality (i.e. feature-set size) of the predicate learning task small enough such that if we find a predicate that fits its training set well, we can be reasonably confident that it will generalize to new instances of the "target class." A lower bound on the number of examples required to PAC-learn some target function $f \in F$ is (Ehrenfeucht, Haussler, Kearns, & Valiant, 1989):

$$m = \Omega\left(\frac{\text{VC-dimension}(F)}{\epsilon}\right)$$

where ϵ is the usual PAC error parameter. We use this bound to get a rough answer to the question:

> *Given m training examples, how large a feature space can we consider such that if we find a promising predicate with our learner in this feature space, we have some assurance that it will generalize well?*

The VC-dimension of a two-class Naive Bayes learner is $n+1$ where n is the number of features. Ignoring constant factors, and solving for n we get Equation 3. Note that this method is only a heuristic. It does not provide any theoretical guarantees about the accuracy of learned clauses since it makes several assumptions (e.g., that the "target function" of the predicate is in F) and does not consider the broader issue of the accuracy of the clause in which the literal will be used.

Another issue is how to set the class priors in the Naive Bayes classifier. Typically, these are estimated by the class frequencies in the training data. These estimates are likely to be biased towards the positive class in our context, however. Consider that estimating the accuracy of a (partially grown) clause by the fraction of positive training-set tuples it covers will usually result in a biased estimate. To compensate for this bias, we simply set the class priors to the uniform distribution. Moreover, when a document does not contain any of the words in the vocabulary of one of our learned classifiers, we assign the document to the negative class (since the priors do not enforce a default class).

Once the training examples and the feature set have been determined, a Naive Bayes model is learned as described in Section 2. The learning task here entails

simply determining the conditional probabilities of words in the vocabulary (i.e. features) given the two classes. By treating this learned model as a Boolean function, we have our candidate Naive-Bayes predicate.

Finally, after the candidate Naive-Bayes predicates are constructed, they are evaluated like any other candidate literal. Those Naive-Bayes predicates that are included in clauses are retained as new background relations so that they may be incorporated into subsequent clauses. Those that are not selected are discarded.

Although our Naive Bayes classifiers produce probabilities for each instance, we do not use these probabilities in our constructed predicates nor in the evaluation of our learned clauses. Naive Bayes' probability estimates are usually poor when its independence assumption is violated, even though its predictive accuracy is often quite good in such situations (Domingos & Pazzani, 1997).

4. Experimental Evaluation

At the beginning of Section 3, we stated that our FOIL-PILFS algorithm has two desirable properties:

- Because it characterizes pages and hyperlinks using a statistical method such as Naive Bayes, its learned rules will not be dependent on the presence or absence of specific key words. Instead, the statistical classifiers used in its learned rules consider the weighted evidence of many words.

- Because it learns each of its statistical predicates to characterize a specific set of pages or hyperlinks, it can perform feature selection in a directed manner. The vocabulary to be used when learning a given predicate can be selected specifically for the particular classification task at hand.

In this section, we test the hypothesis that this approach will learn definitions with higher accuracy than a comparable relational method without the ability to use such statistical predicates. Specifically, we compare our FOIL-PILFS method to ordinary FOIL on several hypertext learning tasks.

4.1. The University Data Set

Our primary data set for these experiments is one assembled for a research project aimed at extracting knowledge bases from the Web (Craven et al., 1998a). This project encompasses many learning problems and we study two of those here. The first is to recognize instances of knowledge base *classes* (e.g. students, faculty, courses etc.) on the Web. In some cases, this can be framed as a page-classification task. We also want to recognize *relations* between objects in our knowledge base. Our approach to this task is to learn prototypical patterns of hyperlink connectivity among pages. For example, a course home page containing a hyperlink with the text `Instructor: Tom Mitchell` pointing to the home page of a faculty member could represent a positive instance of the instructors_of_course relation.

Table 3. Data set distribution per class and per university.

University	student	course	faculty	project	other
Cornell	128	44	34	20	641
Texas	148	38	46	18	577
Washington	126	77	31	21	951
Wisconsin	156	85	42	25	959

Our data set consists of pages and hyperlinks drawn from the Web sites of four computer science departments. This data set includes 4,167 pages and 10,353 hyperlinks interconnecting them. Each of the pages is labeled as being the home page of one of seven classes: course, faculty, student, project, staff, department, and the catch-all other class. For the classification experiments in section 4.3 we use only four of these classes and pool the remaining examples into a single other class. The distribution of examples for each class and for each university is shown in Table 3.

The data set also includes instances of the relations between these entities. Each relation instance 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. The complete data set is available at `http://www.cs.cmu.edu/~WebKB/`.

All of the experiments presented with this data set use *leave-one-university-out* cross-validation, allowing us to study how a learning method performs on data from an unseen university. This is important because we evaluate our knowledge-base extraction system, which this research is a component of, on previously unseen Web sites.

4.2. The Representations

For the experiments in Sections 4.3 and 4.4, we give FOIL the background predicates described in Section 2.2. One issue that arises in using the predicates that represent words in pages and hyperlinks is selecting the vocabulary for each one. For our experiments, we remove *stop-words* and apply the Porter stemming algorithm (Porter, 1980) to the remaining words (refer back to Section 2 for descriptions of these processes). We then use frequency-based vocabulary pruning as follows:

- has_*word* (Page) : We chose words that occur at least 200 times in the training set. This procedure results in 607 to 735 predicates for each training set.

- has_anchor_*word*(Hyperlink) : The vocabulary for this set of relations includes words that occur at least five times among the hyperlinks in the training set. This results in 637 to 738 predicates, depending on the training set.

- has_neighborhood_*word*(Hyperlink): The vocabulary for this set of relations includes words that occur at least twenty times among the hyperlinks in the training set. This set includes 633 to 1025 predicates, depending on the training set.

The FOIL-PILFS algorithm is given as background knowledge the relations listed in Section 2.2, *except for* the three relations above. Instead, it is given the ability to invent predicates that describe the words in pages and the anchor and neighboring text of hyperlinks. Effectively, the two learners have access to the same information as input. The key difference is that whereas ordinary FOIL is given this information in the form of background predicates, we allow FOIL-PILFS to reference page and hyperlink words only via invented Naive-Bayes predicates.

4.3. Experiments in Learning Page Classes

To study page classification, we pick the four largest classes (excluding other) from our university data set: student, course, faculty and project. Each of these classes in turn is the positive class for a binary page classification task. Pages from the remaining classes (staff, department and other) were pooled into a new other class and were present in the negative class for all four classification tasks, along with the pages for the three classes not being learned. For example, we learn a classifier to distinguish student home pages from all other pages. We run FOIL and FOIL-PILFS on these tasks, as well as a Naive Bayes classifier applied directly to the pages.

For each classifier used here, we can associate a numerical confidence along with each prediction. For Naive Bayes, these confidence measures come from the predicted probabilities of class membership. For the relational methods, the confidence measure for a given example is the estimated accuracy of the first clause that the example satisfies.[1] We estimate the accuracy of each of our learned clauses by calculating an m-estimate (Cestnik, 1990) 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 $m = 5$ 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.[2]

By varying a threshold at which we accept a positive prediction, we can trade off precision ("quality" of positive predictions) for recall ("quantity" of positive predictions), depending on the target system requirement for these predictions. Recall and precision are defined as follows:

$$\text{Recall} = \frac{\text{\# correct positive predictions}}{\text{\# positive examples}}$$

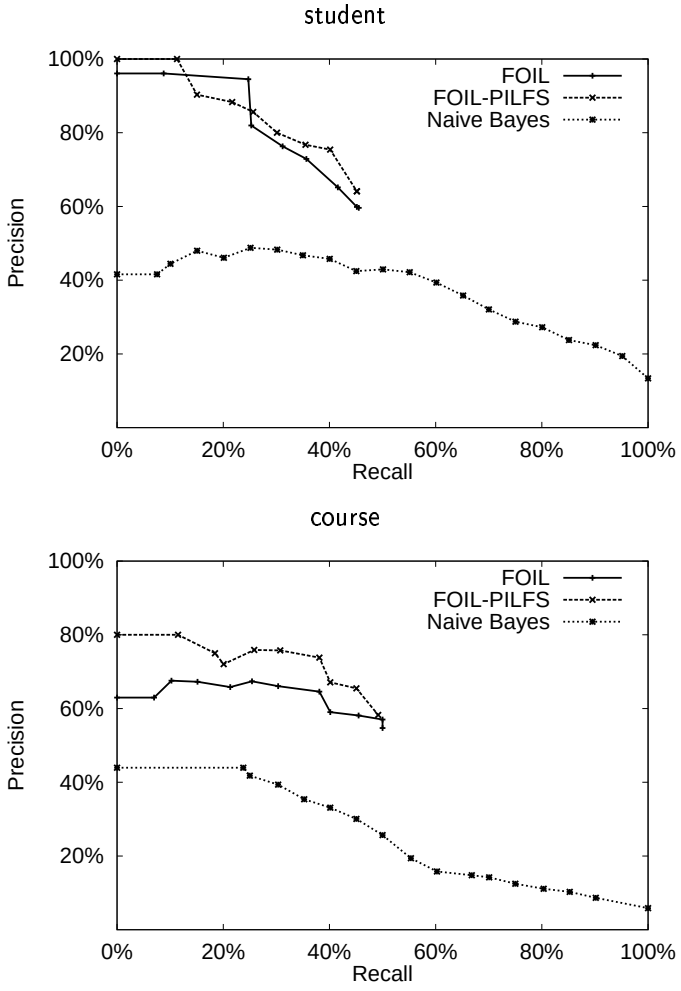

Figure 1. Recall-Precision curves for each algorithm on the student and course classification tasks.

$$\text{Precision} = \frac{\#\text{ correct positive predictions}}{\#\text{ positive predictions}}$$

Plotting recall against precision at various thresholds give us recall-precision curves such as those for our four page-classification tasks shown in Figures 1 and 2. Each point on the recall precision plot represents a particular classifier with that recall and precision performance, obtained by using the threshold used to determine the classifier.

Looking at the graphs in Figures 1 and 2, we note that neither FOIL or FOIL-PILFS can match Naive Bayes for recall performance in the limit. Since Naive Bayes

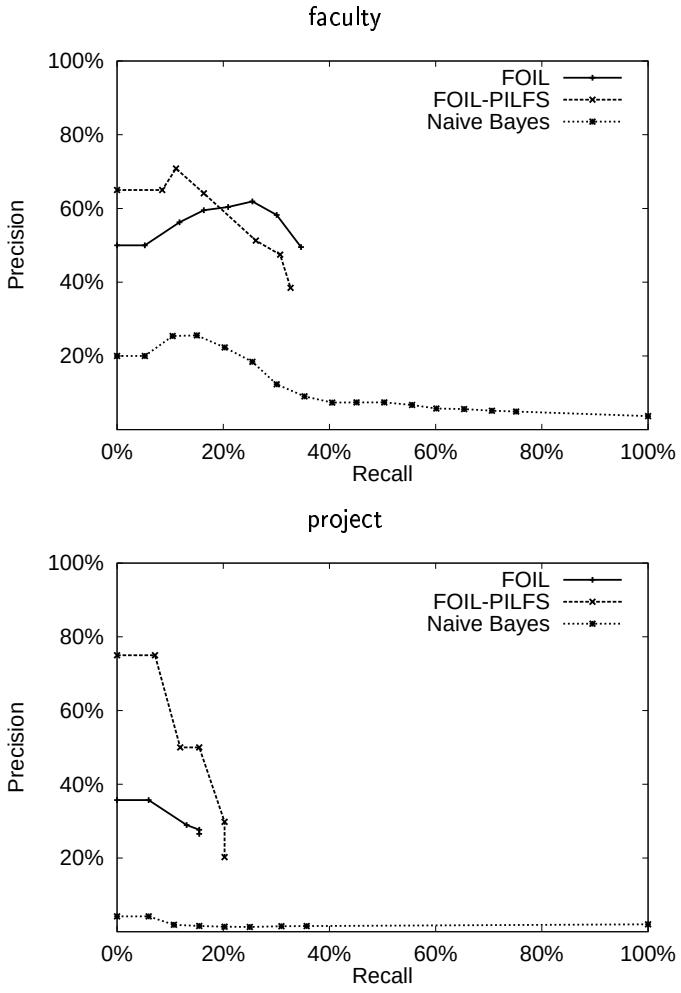

Figure 2. Recall-Precision curves for each algorithm on the faculty and project classification tasks.

is combining evidence from whatever features it observes in a given test example, it can classify an example as positive if we lower the threshold enough. The relational algorithms, on the other hand, need a test example to match all the conditions in some learned rule before a prediction and associated confidence can be obtained.

In contrast, both FOIL and FOIL-PILFS achieve relatively high precision at the lower recall rates they are confined to. Previous research (Craven et al., 1998b) has shown that the relational approach to classifying hypertext often results in better classifier precision. For two of the classes, student and faculty, the recall-

Table 4. Recall (R), precision (P) and F_1 scores on each of the classification tasks for Naive Bayes, FOIL, and FOIL-PILFS

	student			course			faculty			project		
method	R	P	F_1	R	P	F_1	R	P	F_1	R	P	F_1
Naive Bayes	52.1	42.3	46.7	46.3	29.6	36.1	22.2	20.1	21.1	1.2	16.7	2.2
FOIL	45.5	59.6	51.6	50.0	57.0	53.3	34.6	49.5	40.8	15.5	27.7	19.9
FOIL-PILFS	46.2	65.5	54.2	53.3	52.6	53.0	36.0	55.0	43.5	27.4	27.7	27.5

Table 5. Pairwise comparison of the classifiers. For each pairing, the number of times one classifier performed better than the other on recall (R) and precision (P) is shown.

	R wins	P wins		R wins	P wins		R wins	P wins
Naive Bayes	5	3	Naive Bayes	4	0	FOIL	6	7
FOIL	10	13	FOIL-PILFS	11	16	FOIL-PILFS	9	9

precision curves are fairly similar. For the other two classes, course and project, the FOIL-PILFS curve is superior to the FOIL curve.

In order to compare the algorithms in more detail, we can look at their recall and precision performance as classifiers. For Naive Bayes in this case, we treat a prediction as positive when the predicted probability for the positive class *geq* 0.5. For FOIL and FOIL-PILFS, we use all of the learned clauses in a given definition as the classifier. Table 4 shows the recall and precision figures for our classifier predictions.[3] Also shown is the F_1 score (van Rijsbergen, 1979; Lewis, Schapire, Callan, & Papka, 1996) for each algorithm on each task. This is a score commonly used in the information-retrieval community which weights precision and recall equally and has nice measurement properties. It is defined as:

$$F_1 = \frac{2PR}{P+R}$$

Comparing the F_1 scores first, we see that both FOIL and FOIL-PILFS outperform Naive Bayes on all tasks. More importantly, we observe that our new combined algorithm outperforms FOIL on three of the four classification tasks.

Comparing the precision and recall results for FOIL and FOIL-PILFS we see that FOIL-PILFS has better recall than FOIL for all four data sets. in all but two cases FOIL-PILFS outperforms FOIL. The increased recall performance is not surprising, given the statistical nature of the predicates being produced. They test the aggregate distribution of words in the test document (or hyperlink), rather than depending on the presence of distinct keywords. Looking at the precision results, there is no clear winner between FOIL and FOIL-PILFS.

Pairwise comparisons of the three algorithms are shown in Table 5. Here we see, for each pair of learning methods, how often one of them outperformed the other on one of the cross validation runs. For example, of the 16 cross validation

```
project_page(A) :- page_naive_bayes_1(A), link_to(B,A,C,D), anchor_naive_bayes_1(B),
                   anchor_naive_bayes_2(B), page_naive_bayes_2(C),
                   page_naive_bayes_3(C).
```

page_naive_bayes_1:	shore, queri, project, object, sequenc, group, date, relat, releas, research ...
anchor_naive_bayes_1:	neural, utc, group, wind, tunnel, network, multiscalar, home, ...
anchor_naive_bayes_2:	neural, utc, home, wind, tunnel, back, multiscalar, paradyn, ...
page_naive_bayes_2:	databas, utexa, problem, mail, version, email, educ, orient, www, contact.
page_naive_bayes_3:	comput, univers, page, scienc, inform, home, depart.

Figure 3. Clause learned by FOIL-PILFS for the project class. This clause covers 33 positive and no negative training examples. On the unseen test set, it covers 7 project pages and 1 non-project page. Also shown are the words with the greatest positive log-odds ratios for each invented predicate.

runs performed, FOIL had better recall than Naive Bayes 10 times, and had better precision 13 times. Confirming the results using the F_1 score above, we see that FOIL-PILFS does indeed seem to outperform FOIL in general , and FOIL outperforms Naive Bayes on all four tasks.

Figure 3 shows a sample clause learned by FOIL-PILFS. This clause uses five invented predicates, one which tests the distribution of words on the page to be classified (A), two that test the distribution of words on an outgoing link (B), and two that test the distribution on the page pointed to by that link (C). Also shown are the words most highly weighted by each of the predicates. These are determined by assessing

$$\log\left(\frac{\Pr(w_i|pos)}{\Pr(w_i|neg)}\right) \tag{4}$$

for each word w_i, where *pos* represents the positive class with respect to the Naive Bayes classifier, and *neg* represents the negative class. Note that for the page_naive_bayes_2 and the page_naive_bayes_3 predicates, all of the words with positive log-odds ratios in their respective models are listed.

4.4. Experiments in Learning Page Relations

In this section we consider learning target concepts that represent specific relations between pairs of pages. We learn definitions for the three relations described in Section 4.1. In addition to the positive instances for these relations, each data set includes approximately 300,000 negative examples. Our experiments here involve one additional set of background relations: *class*(Page). For each *class* from the previous section, the corresponding relation lists the pages that represent instances of

Table 6. Recall (R), precision (P) and F_1 results for the relation learning tasks.

	department_of_person			instructors_of_course			members_of_project		
method	R	P	F_1	R	P	F_1	R	P	F_1
PATH-FOIL	49.3	84.8	62.4	66.9	74.7	70.6	56.1	73.1	63.5
PATH-FOIL-PILFS	75.8	98.4	85.7	60.6	86.9	71.4	55.4	81.0	65.8

Table 7. Recall (R) and precision (P) results for the relation learning tasks.

	department_of_person		instructors_of_course		members_of_project	
method	R wins	P wins	R wins	P wins	R wins	P wins
PATH-FOIL	1	1	2	0	2	0
PATH-FOIL-PILFS	2	2	1	2	2	4

class. These instances are determined using actual classes for pages in the training set and predicted classes for pages in the test set.

As in the previous section, we learn the target concepts using both (i) a relational learner given background predicates that provide a bag-of-words representation of pages and hyperlinks, and (ii) a version of our FOIL-PILFS algorithm. The base algorithm we use here is slightly different than FOIL, however.

In previous work, we have found that FOIL's hill-climbing search is not well suited to learning these relations for cases in which the two pages of an instance are not directly connected. Thus, for the experiments in this section, we augment both algorithms with a deterministic variant of Richards and Mooney's *relational pathfinding* method (Richards & Mooney, 1992). The basic idea underlying this method is that a relational problem domain can be thought of as a graph in which the nodes are the domain's constants and the edges correspond to relations which hold among constants. The algorithm tries to find a small number of prototypical paths in this graph that connect the arguments of the target relation. Once such a path is found, an initial clause is formed from the relations that constitute the path, and the clause is further refined by a hill-climbing search.

Also, like Džeroski and Bratko's m-FOIL (Džeroski & Bratko, 1992), both algorithms considered here use m-estimates of a clause's error to guide its construction. We have found that this evaluation function results in fewer, more general clauses for these tasks than FOIL's information gain measure.

As in the previous experiment, the only difference between the two algorithms we compare here is the way in which they use predicates to describe word occurrences. We refer to the baseline method as PATH-FOIL, and we refer to the variant of FOIL-PILFS used here as PATH-FOIL-PILFS. We do not consider directly applying the Naive Bayes method in these experiments since the target relations are of arity two and necessarily require a relational learner.

Table 6 shows recall, precision, and F_1 results for the three target relations. For department_of_person, PATH-FOIL-PILFS provides significantly better recall and

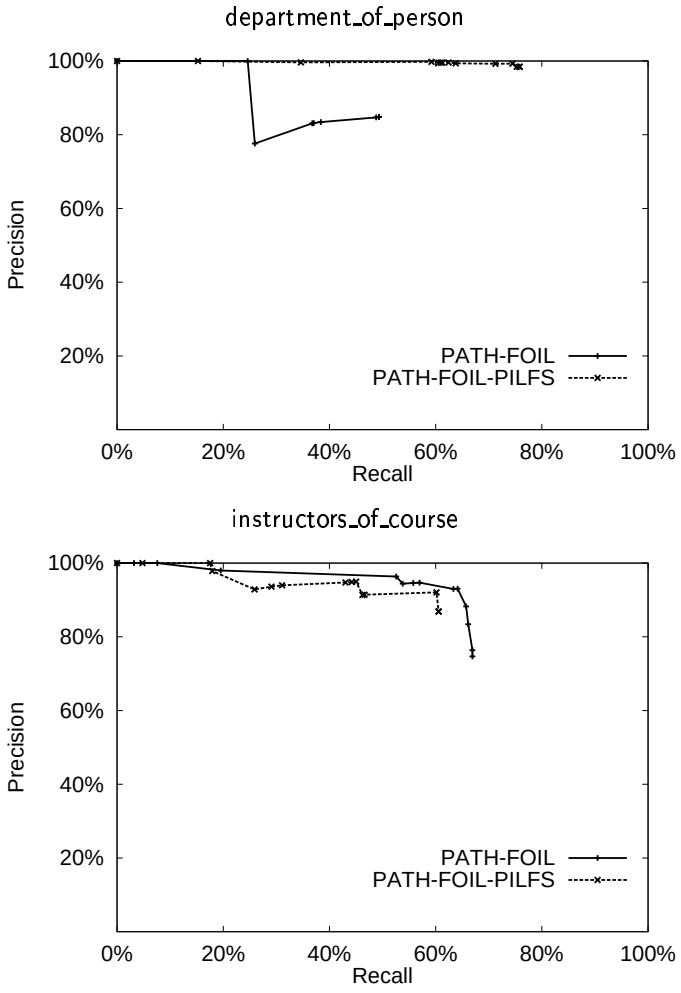

Figure 4. Recall-Precision curves for the department_of_person and instructors_of_course relation learning tasks.

precision than PATH-FOIL. For the other two target concepts, PATH-FOIL has better precision, but slightly worse recall. PATH-FOIL-PILFS has superior F_1 scores for all three target relations. Table 7, shows the number of cross-validation folds for which one algorithm outperformed another. As this table shows, PATH-FOIL-PILFS is the clear winner in terms of precision, but that the results are mixed for recall.

Figures 4 and 5 show the recall-precision curves for the three page-relation tasks. These curves suggest that, whereas PATH-FOIL is perhaps better for instructors_of_course,

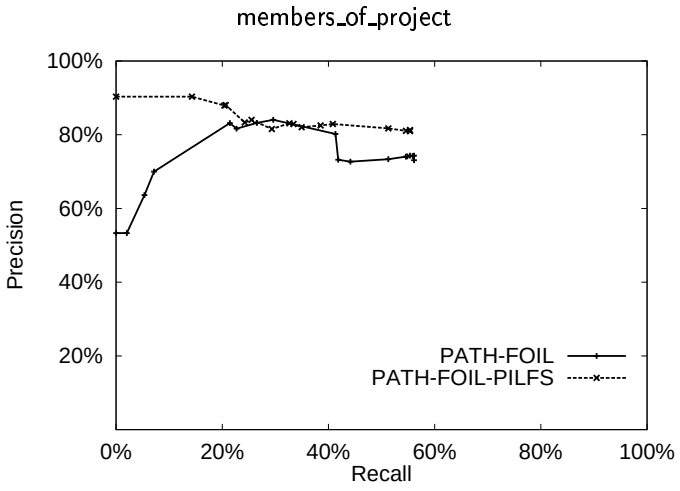

Figure 5. Recall-Precision curves for the members_of_project task.

PATH-FOIL-PILFS is the clear winner on the department_of_person, and superior for members_of_project as well.

4.5. Relational Learning and Internal Page Structure

So far we have considered relational learning applied to tasks that involve representing the relationships *among* hypertext documents. Hypertext documents, however, have internal structure as well. In this section we apply our learning method to a task that involves representing the internal layout of Web pages. Specifically, the task we address is the following: given a reference to a country name in the Web page of a company, determine if the company has operations in that country or not.

Our approach makes use of an algorithm that parses Web pages into tree structures representing the layout of the pages (DiPasquo, 1998). For example, one node of the tree might represent an HTML table where its ancestors are the HTML headings that come above it in the page. In general, any node in the tree can have some text associated with it. We frame our task as one of classifying nodes that contain a country name in their associated text.

In our experiments here we apply FOIL and FOIL-PILFS to this task using the following background relations:

- heading(Node, Page), li(Node, Page), list(Node, Page), list_or_table(Node, Page), paragraph(Node, Page), table(Node, Page), td(Node, Page), title(Node, Page), tr(Node, Page): These predicates list the nodes of each given type, and the page in which a node is contained. The types correspond to HTML elements.

- ancestor(Node, Node), parent(Node, Node), sibling(Node, Node), ancestor_heading(Node, Node), parent_heading(Node, Node): These predicates represent relations that hold among the nodes in a tree. The two relations ancestor_heading and parent_heading are specializations of ancestor and heading, respectively. They are used to relate given nodes to the HTML heading tags that are their ancestors or direct parents.

The target relation, has_location(Node, Page), is a binary relation so that the learner can easily relate nodes by their common page as well as by their relationship in the tree. In a setup similar to our previous experiments, we give FOIL a set of has_node_*word*(Node) predicates, and we allow FOIL-PILFS to invent predicates that characterize the words in nodes. Our data set for this task consists of 788 pages parsed into 44,760 nodes. There are 337 positive instances of the target relation and 363 negative ones. We compare FOIL to FOIL-PILFS on this task using a five-fold cross-validation run.

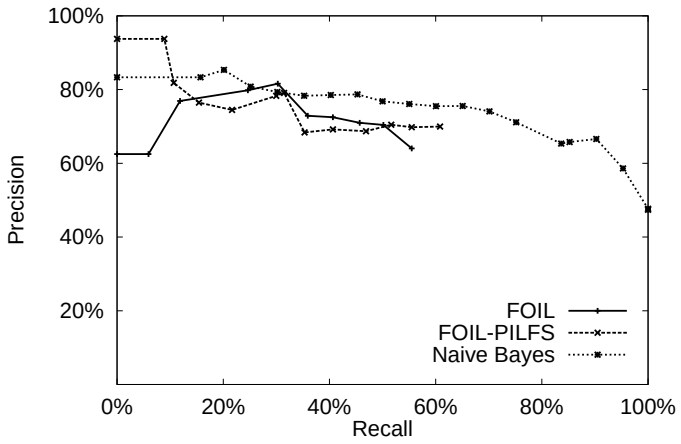

Figure 6. Recall-Precision curves for each algorithm on the node classification task.

Figure 6 shows the recall-precision curve for the three algorithms on this task. In this case, the relational representation is only a win at the lower recall levels where, as before, we get better precision performance than Naive Bayes. The performance of FOIL and FOIL-PILFS on this graph is fairly comparable, except at low recall levels.

Table 8 shows the recall, precision and F_1 results for our algorithms viewed as simple classifiers. Additionally, the table shows the number of folds for which one algorithm outperformed the other in terms of precision or recall. FOIL-PILFS provides better recall at the cost of slightly worse precision than ordinary FOIL under this evaluation. As expected, FOIL-PILFS outperformed FOIL on recall in four of the five folds. It produced better precision results in three of the five folds.

Table 8. Recall (R), precision (P), and F_1 results for the node classification task.

method	R	P	F_1	R wins	P wins
FOIL	55.5	64.0	59.5	1	2
FOIL-PILFS	61.4	63.1	62.2	4	3

Table 9. Recall (R), precision (P) and F_1 scores for FOIL-PILFS on the four page classification tasks as we vary ϵ.

	student			course			faculty			project		
ϵ	R	P	F_1	R	P	F_1	R	P	F_1	R	P	F_1
0.01	43.4	71.2	53.9	57.4	54.3	55.8	36.0	44.7	39.9	19.1	28.6	22.9
0.05	45.2	64.1	53.0	49.2	58.8	53.6	32.7	40.3	36.1	20.2	29.8	24.1
0.10	46.2	65.5	54.2	53.3	52.6	53.0	36.0	55.0	43.5	27.4	27.7	27.5
0.15	34.0	61.7	43.9	55.7	54.4	55.1	37.3	58.2	45.4	19.1	21.3	20.1
0.20	41.4	61.8	49.6	53.7	58.5	56.0	34.6	40.8	37.5	14.3	26.7	18.6

4.6. Varying the Vocabulary Parameter in FOIL-PILFS

As described in Section 3, our FOIL-PILFS algorithm employs a parameter, ϵ, which controls how many words Naive Bayes can use when constructing a new predicate. In contrast to our experiments with ordinary FOIL, where we had to make vocabulary-size decisions separately for the page, anchor and neighborhood predicates, ϵ provides a single parameter to set when using FOIL-PILFS.

In all of our experiments so far we have set $\epsilon = 0.1$. In order to assess how FOIL-PILFS's performance is affected by varying ϵ, we rerun the page classification experiment from Section 4.3 with ϵ set to 0.01, 0.05, 0.15 and 0.2. The smaller ϵ forces Naive Bayes to work with fewer words, the larger allows it up to twice as many as in our original experiments. Precision, recall and F_1 scores for this experiment are shown in Table 9.

It is hard to see general trends in this table. We note, however, that most of the F_1 values in this table are superior to the corresponding F_1 values for FOIL shown in Table 4. This result suggests that the outcomes of our previously described experiments did not depend on a fortuitous choice of ϵ. Finally, we note that this single parameter to be set is preferable to the case of ordinary FOIL where we had to set three different vocabulary-size parameters.

5. Related Work

The idea of predicate invention has a long history in the field of inductive logic programming; there are several reviews of work done in this area (Stahl, 1996; Kramer, 1995). Our FOIL-PILFS method is similar to the CHAMP (Kijsirikul et al., 1992), CWS (Srinivasan, Muggleton, & Bain, 1992), MOBAL (Wrobel, 1994), and

CHILLIN (Zelle, Mooney, & Konvisser, 1994) systems which all invent predicates to cover an extensionally given set of target tuples. Srinivasan and Camacho (Srinivasan & Camacho, 1999) have also developed an algorithm that combines a relational learner with a numeric feature-value learner. FOIL-PILFS differs from these systems in the method it uses to define new predicates (Naive Bayes), and in its policy of liberally considering new invented predicates. It was designed with the special properties of text and hypertext in mind.

Our work is also related to recent research on learning *probabilistic relational models* (Koller & Pfeffer, 1997; Friedman, Getoor, Koller, & Pfeffer, 1999). There are several key differences, however. Whereas we have focused on learning predictive models for particular target concepts, the probabilistic relational approach focuses on the more general task of learning a joint probability distribution over the relevant features in a problem domain. Moreover, whereas our approach can use an existentially quantified variable to characterize some entity related to another entity of interest, the probabilistic relational approach can characterize only aggregate properties of related entities Finally, the probabilistic relational approach has not yet been applied to large, complex data sets as FOIL-PILFS has.

6. Conclusions

We have presented a hybrid relational/statistical approach to learning in text domains. Whereas the relational component is able to describe the graph structure of hyperlinked pages and the internal structure of HTML pages, the statistical component is adept at learning predicates that characterize the distribution of words in pages, hyperlinks and parts of pages. We described one particular instantiation of this approach: an algorithm based on FOIL that invents predicates on demand which are represented as Naive Bayes models. We evaluated this approach by comparing it to a baseline method that represents words directly in background relations. Our experiments indicate that our method generally learns more accurate definitions.

This work has explored one particular method for combining relational and statistical learning. Currently, we are exploring a number of directions in this general framework:

- Investigating other search strategies. Because FOIL's hill-climbing search is myopic, we suspect that it does not add literals describing relationships among documents as often as it would be profitable to do so. One modification to the search strategy that we are currently investigating is the use of *relational cliches* (Silverstein & Pazzani, 1991). Relational cliches consist of sequences of predicates to be considered in a single search step.

- Using the confidence scores produced by our invented Naive Bayesian predicates. Currently we treat our Naive Bayes models as Boolean predicates by thresholding on confidence ≥ 0.5. We are investigating methods that use these probability estimates to combine evidence across the literals of a clause.

- Simultaneously fitting all of the parameters in a clause. The FOIL-PILFS approach involves incrementally adding statistical predicates to a clause in a hill-

climbing search. We hypothesize that clauses with more globally optimal combinations of literals can be learned by simultaneously trying to learn all of the predicates in a clause. Specifically, we are investigating an approach that views the predicates as hidden variables and uses the Expectation Maximization (EM) algorithm to determine the parameters of all of the predicates at once.

Finally, we believe that our approach is applicable to learning tasks other than those that involve hypertext. We hypothesize that it is well suited to other domains that involve both relational structure, and potentially large feature spaces. In future work, we plan to apply our method in such domains.

Acknowledgments

Thanks to Dan DiPasquo for his assistance with the experiments reported in Section 4.5, to Tom Mitchell for many insightful comments, and to Nicolas Lachiche for pointing out a problem with an earlier version of our data set. This research was conducted at Carnegie Mellon University and supported in part by the DARPA HPKB program under contract F30602-97-1-0215.

Notes

1. This method for estimating confidence of a prediction on a test example was chosen for ease of implementation and because of its close relation to how FOIL classifies test examples. Of course it ignores available information about the other rules that matched the test example.
2. This change does not affect the classifications made by a learned set of clauses. It affects only our confidence associated with each prediction.
3. Since our graphs show the best precision at a given recall, the endpoint precision values in the graph may be slightly higher than those in our tables.

References

Cestnik, B. (1990). Estimating probabilities: A crucial task in machine learning. In *Proceedings of the Ninth European Conference on Artificial Intelligence*, pp. 147–150 Stockholm, Sweden. Pitman.

Cohen, W. W. (1995a). Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 115–123 Tahoe City, CA. Morgan Kaufmann.

Cohen, W. W. (1995b). Learning to classify English text with ILP methods. In Raedt, L. D. (Ed.), *Advances in Inductive Logic Programming*. IOS Press.

Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., & Slattery, S. (1998a). Learning to extract symbolic knowledge from the World Wide Web. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pp. 509–516 Madison, WI. AAAI Press.

Craven, M., Slattery, S., & Nigam, K. (1998b). First-order learning for Web mining. In *Proceedings of the Tenth European Conference on Machine Learning*, pp. 250–255 Chemnitz, Germany. Springer-Verlag.

DiPasquo, D. (1998). Using HTML formatting to aid in natural language processing on the World Wide Web.. Senior thesis, Computer Science Department, Carnegie Mellon University.

Domingos, P., & Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, *29*, 103–130.

Džeroski, S., & Bratko, I. (1992). Handling noise in inductive logic programming. In *Proceedings of the Second International Workshop on Inductive Logic Programming*, pp. 109–125 Tokyo, Japan.

Ehrenfeucht, A., Haussler, D., Kearns, M., & Valiant, L. (1989). A general lower bound on the number of examples needed for learning. *Information and Computation*, *82*(3), 247–251.

Friedman, N., Getoor, L., Koller, D., & Pfeffer, A. (1999). Learning probabilistic relational models. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pp. 1300–1307 Stockholm, Sweden. Morgan Kaufmann.

Joachims, T., Freitag, D., & Mitchell, T. (1997). WebWatcher: A tour guide for the World Wide Web. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pp. 770–775. Morgan Kaufmann.

Kijsirikul, B., Numao, M., & Shimura, M. (1992). Discrimination-based constructive induction of logic programs. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pp. 44–49 San Jose, CA. AAAI Press.

Koller, D., & Pfeffer, A. (1997). Learning probabilities for noisy first-order rules. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pp. 1316–1321 Nagoya, Japan. Morgan Kaufmann.

Kramer, S. (1995). Predicate invention: A comprehensive view. Tech. rep. OFAI-TR-95-32, Austrian Research Institute for Artificial Intelligence, Vienna, Austria.

Kushmerick, N., Weld, D. S., & Doorenbos, R. (1997). Wrapper induction for information extraction. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pp. 729–737 Nagoya, Japan. Morgan Kaufmann.

Lewis, D. D., & Ringuette, M. (1994). A comparison of two learning algorithms for text categorization. In *Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval*, pp. 81–93. ISRI; Univ. of Nevada, Las Vegas.

Lewis, D. D., Schapire, R. E., Callan, J. P., & Papka, R. (1996). Training algorithms for linear classifiers. In *Proceedings of the Nineteenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pp. 298–306 Zurich, Switzerland. ACM.

Mitchell, T. (1997). *Machine Learning*. McGraw Hill.

Mladenić, D. (1996). Personal WebWatcher: Design and implementation. Tech. rep. IJS-DP-7472, Department for Intelligent Systems, J. Stefan Institute, Ljubljana, Slovenia.

Moulinier, I., Raškinis, G., & Ganascia, J.-G. (1996). Text categorization: a symbolic approach. In *Proceedings of the 6th Annual Symposium on Document Analysis and Information Retrieval*.

Pazzani, M. J., Muramatsu, J., & Billsus, D. (1996). Syskill & Webert: Identifying interesting Web sites. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp. 54–59 Portland, OR. AAAI Press.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, *14*(3), 130–137.

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

Quinlan, J. R., & Cameron-Jones, R. M. (1993). FOIL: A midterm report. In *Proceedings of the Fifth European Conference on Machine Learning*, pp. 3–20 Vienna, Austria. Springer-Verlag.

Richards, B., & Mooney, R. (1992). Learning relations by pathfinding. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pp. 50–55 San Jose, CA. AAAI Press.

Silverstein, G., & Pazzani, M. J. (1991). Relational clichés: Constraining constructive induction during relational learning. In *Proceedings of the Eighth International Workshop on Machine Learning*, pp. 203–207 Evanston, IL. Morgan Kaufmann.

Soderland, S. (1997). Learning to extract text-based information from the World Wide Web. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pp. 251–254 Newport Beach, CA. AAAI Press.

Srinivasan, A., & Camacho, R. (1999). Numerical reasoning with an ILP system capable of lazy evaluation and customised search. *The Journal of Logic Programming*, *40*(2/3), 185–213.

Srinivasan, A., Muggleton, S., & Bain, M. (1992). Distinguishing exceptions from noise in non-monotonic learning. In *Proceedings of the Second International Workshop on Inductive Logic Programming* Tokyo, Japan.

Stahl, I. (1996). Predicate invention in inductive logic programming. In DeRaedt, L. (Ed.), *Advances in Inductive Logic Programming*. IOS Press.

van Rijsbergen, C. J. (1979). *Information Retrieval*. Butterworths.

Wrobel, S. (1994). Concept formation during interactive theory revision. *Machine Learning*, *14* (2), 169–191.

Yang, Y., & Pedersen, J. (1997). A comparative study on feature set selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 412–420 Nashville, TN. Morgan Kaufmann.

Zelle, J. M., Mooney, R. J., & Konvisser, J. B. (1994). Combining top-down and bottom-up techniques in inductive logic programming. In *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 343–351 Rutgers, NJ. Morgan Kaufmann.