Hyperlink Ensembles: A Case Study in Hypertext Classification

Johannes Fürnkranz
Austrian Research Institute for Artificial Intelligence
Schottengasse 3, A-1010 Wien, Austria
E-mail: juffi@oefai.at

Technical Report OEFAI-TR-2001-30

Abstract

In this paper, we introduce hyperlink ensembles, a novel type of ensemble classifier for classifying hypertext documents. Instead of using the text on a page for deriving features that can be used for training a classifier, we suggest to use portions of texts from all pages that point to the target page. A hyperlink ensemble is formed by obtaining one prediction for each hyperlink that points to a page. These individual predictions for each hyperlink are subsequently combined to a final prediction for the class of the target page. We explore four different ways of combining the individual predictions and four different techniques for identifying relevant text portions. The utility of our approach is demonstrated on a set of Web-pages that relate to Computer Science Departments.

1 Introduction

Ensemble techniques have received considerable attention within the recent machine learning literature [24, 25, 51, 3]. The idea to obtain a diverse set of classifiers for a single learning problem and to vote or average their predictions is both simple and powerful, and the obtained accuracy gains often have a sound theoretical foundation. Averaging the predictions of the individual classifiers helps to reduce the variance and often increase the reliability of the final prediction. There are several techniques for obtaining a diverse set of classifiers. Most common is to use subsampling for diversifying the training sets as in bagging [8] and boosting [30]. Other techniques include the use of different feature subsets [4], to exploit the randomness of the base algorithms [40], possibly by artificially randomizing their behavior [26], or by modifying the output labels [27].

Not surprisingly, the power of ensemble techniques was also recognized for text categorization tasks. For example, Larkey and Croft [42] found that combining the predictions of different classifiers for the same problem improves the results; error-correcting output codes were successfully applied to various text classification problems [37, 5]; and Weiss et al. [65] applied boosted decision trees to the Reuters-21578

benchmark data set and achieved one of the best results reported in the literature. A possible reason for (at least the latter) success lies in the explanation that boosting is another technique for maximizing the margin (i.e., the distance between a decision boundary and the classes it separates [60]). The same principle is at the core of support vector machines, which have also been successfully applied to text categorization tasks [38].

In this paper, we investigate a different way for constructing an ensemble of diverse classifiers, which is specifically tailored for classifying Web documents. The technique is based on the observation that typically one has information from many incoming hyperlinks that point to a page. Our proposal is to exploit this information by training a classifier that predicts the class of a page by first classifying each incoming hyperlink in isolation. This ensemble of predictions is then used to derive the final prediction by looking for the majority among the individual predictions, thereby incorporating confidence information about each prediction. Our results show that in the WEB→KB domain, documents can be classified more reliably with information originating from pages that point to the document rather than with features that are derived from the document text itself.

Some of the results presented in this paper have previously appeared as [32], in a condensed format and without making the connection to ensemble techniques explicit.

2 Motivation

Vast amounts of documents are available on-line, and classifying them into meaningful semantic categories is a rewarding and challenging task. Applications of classification techniques include browsing assistants [44, 2, 53], information filtering [41], e-mail sorting [17, 52, 56], recognizing entities in ontologies [20, 48], and many more [49]. While conventional information retrieval focuses primarily on information that is provided by the text that can be found in the documents, Web documents provide additional information through the way in which different documents are connected to each other by *hyperlinks*. In particular, the information on what we will call *predecessor pages*—pages that have a hyperlink pointing to the *target page*—may contain useful information for classifying a page. Consider the set of pages shown in Figure 1. The target page itself (shown at the bottom) does not contain much information about its author. However, from any of its three predecessor pages can be inferred that the person in question is a professor, and the union of the three pages reveals information about his university, one of his students, and his colleagues.

There are several reasons why the information on predecessor pages can improve the performance of a text classifier for Web pages:

redundancy: Quite often there is more than one page pointing to a single page on the Web. The ability to combine multiple, independent sources of information can improve classification accuracy as is witnessed by the success of ensemble techniques in other areas of machine learning.

independent encoding: As the set of predecessor pages typically originates from several different authors, the provided information is less sensitive to the vocabulary used by one particular author.

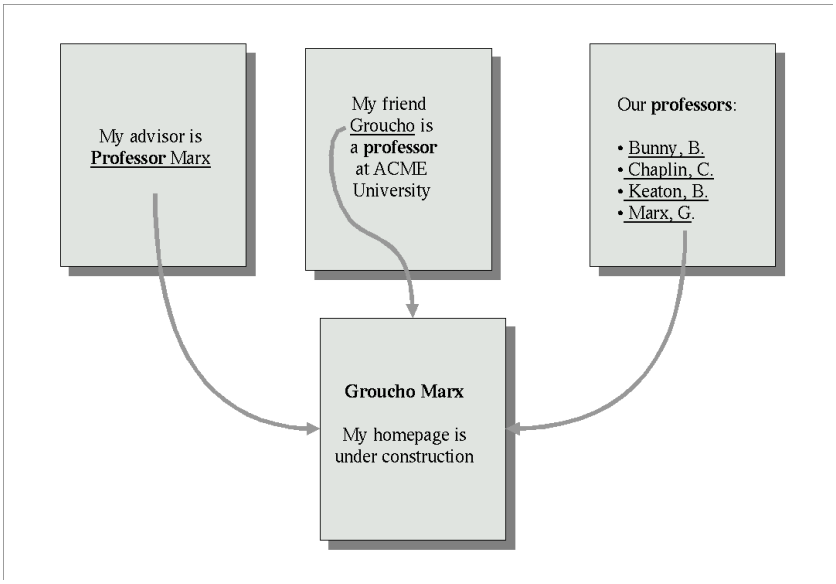

Figure 1: An example for an uninformative target page which can be recognized as a professor's home-page by three predecessor pages.

sparse or non-existing text: Many Web pages only contain a limited amount of text. In fact, many pages only contain images and no machine-readable text at all. Looking at predecessor pages increases the chances of encountering informative text about the page to classify.

irrelevant or misleading text: Often pages contain irrelevant text. The target page in Figure 1 is such an example. In other cases, pages are designed to deliberately contain misleading information. For example, many pages try to include a lot of keywords into comments or invisible parts of the text in order to increase the breadth of their indexing for word-based search engines.[1] Again, the fact that predecessor pages are typically authored by different and independent sources may provide relevant information or improve focus.

foreign language text: English is the predominant language on the Web. Nevertheless, documents in other languages occur in non-negligible numbers. Even if the text on the page is written in a foreign language, there may be incoming links from pages that are written in English. In many cases, this allows an English speaker (or a text classifier based on English vocabulary) to infer something about the contents of the page.

[1] For example, the author once encountered a page of an Austrian ski ressort that contained all names of its competitors in comments in order to be retrieved if somebody looks for information of one of its competitors (in fact this was how the page was found by the author). For recognizing this page as a ski ressort, this information may be helpful. On the other hand, the author also came across a page that included an entire dictionary of the most common words of the English language as invisible text…

In summary, the use of text on predecessor pages may provide a richer vocabulary, independent assessment of the contents of a page through multiple authors, redundancy in classification, focus on important aspects, and a simple mechanism for dealing with pages that have sparse text, no text, or text in different languages.

3 Related Work

The importance of information contained in the hyperlinks pointing to a page has been recognized early on. Anchor texts (texts on hyperlinks in an HTML document) of predecessor pages were already indexed by the World-Wide Web Worm, one of the first search engines and Web crawlers [46]. Spertus [64] suggests a taxonomy of different types of (hyper-)links that can be found on the Web and discusses how the links can be exploited for various information retrieval tasks on the Web. Lately, the study of the graph properties of the Web has increased considerably [10]. This is mostly due to the success of algorithms that explicitly focus on the exploitation of the link structure of the Web. Prominent examples are the HITS algorithm for determining good hub and authority pages [39] and the PageRank algorithm for approximating the probability that a page is visited by a random surfer on the Web [9], which have been successfully applied to various web mining tasks [13, 23, 7]. Prerequisite for this type of research is the availability of extended document indexing techniques that allow fast access to both outgoing and incoming hyperlinks of a page [6]. An excellent overview of recent research in the area, which is nowadays referred to as *web mining*, can be found in [11].

Not surprisingly, the potential of hyperlinks as additional information sources for text categorization tasks has also been looked at in recent research. Previous work on hypertext categorization in one way or another addressed this problem by merging (parts of) the text of the predecessor pages with the text of the page to classify or by keeping a separate feature set for the predecessor pages. For example, Chakrabarti et al. [12] evaluated two variants, one that simply appends the text of the neighboring (predecessor and successor) pages to the text of the target page, and one that uses two different sets of features, one for the target page and one for a concatenation of the neighboring pages. The results were negative: in two domains both approaches performed worse than the conventional technique that uses only features of the target document. From these results, Chakrabarti et al. [12] concluded that the text from the neighbors is too noisy to help classification and proposed a different technique that included predictions for the class labels of the neighboring pages into the computation of the prediction for the target page. Unless the labels for the neighbors are known a priori, the implementation of this approach requires an iterative technique for assigning the labels, because changing the class of a page may potentially change the class assignments for all neighboring pages as well. The authors implemented a relaxation labeling technique, and showed that it improves performance over the standard text-based approach that ignores the hyperlink structure. The utility of classes predictions for neighboring pages was confirmed by the results of Oh et al. [50] and Yang et al. [68].

A different line of research concentrates on explicitly encoding the relational structure of the Web in first-order logic. For example, a binary predicate `link_to(page1,`

`page2)` can be used to represent the fact that there is a hyperlink on `page1` that points to `page2`. In order to be able to deal with such a representation, one has to go beyond traditional attribute-value learning algorithms and resort to inductive logic programming. Craven et al. [22] use a variant of Foil [54] to learn classification rules that can incorporate features from neighboring pages. The algorithm uses a deterministic version of relational path-finding [57], which overcomes Foil's restriction to determinate literals [55], to construct chains of `link_to/2` predicates that allow the learner to access the words on a page via a predicate of the type `has_`*`word`*`(page)`. For example, the conjunction `link_to(P1,P), has_`*`word`*`(P1)` means "there exists a predecessor page `P1` that contains word *word*. Slattery and Mitchell [63] improve the basic Foil-like learning algorithm by integrating it with ideas originating from the HITS algorithm for computing hub and authority scores of pages [39], while Craven and Slattery [21] combine it favorably with a Naive Bayes classifier.

At its core, using features of pages that are linked via a `link_to/2` predicate is identical to the approach evaluated by Chakrabarti et al. [12] where words of neighboring documents are added as a separate feature set: in both cases, the learner has access to all the features in the neighboring documents. The main difference lies in the fact that in the relational representation, the learner may control the depth of the chains of `link_to/2` predicates, i.e., it may incorporate features from pages that are several links away. From a practical point of view, the main difference lies in the types of learning algorithms that are used with both approaches: while inductive logic programming typically relies on rule learning algorithms that try to classify pages by forming "hard" classification rules that predict a class by looking only at a few selected features, the approach of [12] is typically used for learning algorithms that always use all of the available features (such as a Naive Bayes classifier). Yang et al. [68] discuss both approaches, relate them to a taxonomy of five possible regularities that may be present in the neighborhood of a target page, and empirically compare them under different conditions.

Our approach differs from these previous works by keeping the features of each individual predecessor page separately. In fact, we propose to use a separate classifier for each predecessor page, which uses features derived from the predecessor to predict the class of the target page. The text of the target page itself is completely ignored. Thus we obtain several predictions for the class of the target page, one prediction for each predecessor page. These predictions can then be combined into a final prediction using simple voting techniques. The next section describes this technique in more detail and discusses its advantages over the above-mentioned approaches.

4 Hyperlink Ensembles

As motivated in Section 2, we believe that the pages that point to a page that we want to classify—the *predecessor pages* of the *target page*—often contain more and better information about the page than the page itself. Several other researchers have based their work on similar hypotheses but with mostly negative results. We believe that the main reason for this is based on two shortcomings of previous works:

- the features of the predecessor pages need to be kept separately
- the redundancy provided by multiple incoming links is important information

Note that in conventional attribute-value representations, it is not possible to represent such a learning problem in a way that features from different predecessors are encoded separately. The reason is that there is an arbitrary number of predecessor pages and there is no clear order of the predecessor pages that would allow an encoding like "this is a feature of the 1st predecessor page" etc.[2]

It is also worth noting that the first two of the above shortcomings are complementary: The idea of concatenating the text of the predecessor pages (as, e.g., used in [12, 50, 68]) maintains the redundancy (at least if a bag-of-word representation is used, which allows the learner to access word frequencies) but it does not keep the features separately. On the other hand, approaches that use a relational representation [22, 21] are in principle able to keep their feature sets separately (one would only have to add an inequality predicate to the background knowledge that allows to compare page identifiers), but the rule-base learning algorithms that are typically used with such a representation construct minimal rule sets that try to make the learned rule sets as concise as possible and do not allow for redundancy.

The approach we propose solves these problems by making a separate prediction for each predecessor page and combining these predictions at classification time. This guarantees that the feature sets for different predecessors are treated differently (each one corresponds to a different classification problem), but that their redundancy is nevertheless exploited (in a way, each predecessor page is asked for its opinion and their cumulative vote is used as the final prediction).

To achieve this, we represent a target page with a set of instances, one for each predecessor page. Each instance consists of features derived from the corresponding predecessor page and the class label of the target page. Let us assume for the moment that we only consider the words on the anchor text as the features that represent the predecessor page. The home page of professor Marx at the bottom of Figure 1 would then be presented with three instances, each represented by the words on the anchor text of the corresponding predecessor page ("professor marx" in the first case, "groucho" for the second page, and "marx g" for the third page). Each of these instances is labeled with the class information of the target page (*professor*).

The data set that is constructed in this way is subsequently used for training a single classifier. This classifier is then able to predict a class label for links between a predecessor page and its corresponding target page. Ideally, all predictions that involve the same target page would uniformly predict the same class. However, in many cases this learned classifier will be imperfect and brittle, so that conflicting predictions are possible or even likely. Hence, we have to resort to some form of voting in order to determine the final prediction (see below). Obviously, this approach may be viewed as an ensemble technique, where the members of the ensembles correspond to different

[2] In principle, this problem is a case of multi-instance learning [28]. We have not explored this relation further, but it should be noted that many algorithms that are used to solve multi-instance learning problems are assuming that it is sufficient to select the best of all different representations of a given instance. This framework corresponds to the *Max* combiner method discussed below. It is interesting to note that this setting performed best in our experiments.

representations of the same problem, one for each hyperlink. Consequently we refer to this technique as *hyperlink ensembles*.

A crucial component of each ensemble technique is the voting procedure that is used for combining the predictions of each member of the ensemble into a final prediction. Proposals range from simple voting to meta-learning techniques like stacking [66], arbiters and combiners [14] or grading [62]. Meta-learning techniques are not straight-forwardly applicable to our approach, as they typically depend on having a fixed number of classifiers in the ensembles (for constructing the meta training set). Several studies have compared voting techniques for combining the predictions of the individual classifiers of an ensemble in various contexts [e.g., 45, 1]. It seems to be the case that techniques that include confidence estimates of the ensemble predictors into the computation of the final predictions are preferable [cf. also 61], but the final word on that issue remains to be spoken. Consequently, we decided to investigate several different approaches for combining classifiers, which include voting, weighted sums, and using the prediction with the highest confidence measure. These approaches are described in detail in Section 5.2.

Similarly, a crucial component of our technique is the representation of the predecessor pages. Previous works used the entire text of the predecessor pages. However, this simple and straight-forward solution is problematic because different parts of a page may be on different topics, some of which may have nothing to do with the target page. In particular, a page with n outgoing links may, in the worst case, be predecessor for n different classes. As we use the same classifier for each predecessor, it will assign the same label to two target pages that happen to have the same predecessor. Even though the other predecessors of each target page may over-rule this assignment, this is clearly not an optimal solution. Consequently, we believe it is necessary to restrict the feature set to the part of the text that is relevant for the hyperlink so that the feature set of the same predecessor page may be different for different target pages. The most obvious choice is to simply use the anchor text that appears on the hyperlink. In our first experiments, we will concentrate on this approach, but other heuristics will be evaluated later in the paper (Section 7).

5 Implementation and Experimental Setup

5.1 The Ripper Rule Learner

We evaluated our approach using the Ripper rule learning algorithm [16], which is available for research purposes from `http://www.research.att.com/~diane/ripperd.html`. Ripper is an efficient, noise-tolerant rule learner based on the incremental reduced error pruning algorithm [36, 31]. Before learning a rule, Ripper splits the available training data into a growing set (usually $2/3$ of the data) and pruning set (the rest of the data). It then learns single rules by greedily adding one condition at a time (using Foil's information gain heuristic [54]) until the rule no longer makes incorrect predictions on the growing set. Thereafter, the learned rule is simplified by deleting conditions as long as the performance of the rule does not decrease on the pruning set. All examples covered by the resulting rule are then removed from the

training set, and a new rule is learned in the same way until all examples are covered by at least one rule. Thus, Ripper is a member of the family of separate-and-conquer (or covering) rule learning algorithms [33]. In a post-processing phase, Ripper uses a global optimization algorithm that allows it to re-evaluate and modify the learned rules in the context of the other rules.

Ripper is able to deal with text categorization problems via the use *set-valued features* [18]. Basically, Ripper represents a document as a set of words, i.e., it represents a document by the set of words that occur in the document, disregarding their order and their frequencies. For rule learning algorithms, this representation seems to come more natural than the commonly used bag-of-words representation, which represents a document as a multi-set (i.e., a *bag*) of its words, i.e., ignoring their order but not their frequency.[3] However, the straight-forward implementation of a set-of-words representation—encoding the presence or absence of each possible word as a separate Boolean feature—results in a very inefficient encoding of the training examples because much space is wasted for specifying the absence of words in a document. Set-valued features allow to specify a set of values instead of a single value as in conventional attribute-value features. Thus, each document can be represented with a single feature that lists all the words that appear in the document. Ripper then considers conditions of the form *word* $\in$ *feature*. These conditions are evaluated as *true* for a given example, whenever *word* is part of the subset of the values of *feature* for this example.

5.2 Voting schemes

As discussed above, we constructed a training set consisting of one example for each predecessor of a page, labeled with the class of the target page. In the simplest case, the features of this set-valued attribute consisted of the words of the anchor text (cf. Section 7 for alternatives). Ripper was trained on this training set producing a single classifier. We used Ripper with the option `-a unordered` which induces a set of unordered rules, i.e., one set of rules for each class, discriminating this class from all other classes.[4] In order to compute a prediction for a target page, this classifier was applied to each of its predecessor pages, and gave a prediction for each of them. These are then combined via a voting scheme.

Based on previous results from similar studies in ensemble techniques [45, 1, 61], we implemented various voting schemes that make use of confidence information, i.e., that incorporate information about how sure the classifier is about its prediction. As Ripper only returns a single prediction and not a probability distribution over all possible classes, we have to associate a confidence score with each of Ripper's predictions. We decided to use the same confidence score for all examples that are covered by the same rule. For this score, we used the Laplace-estimate $\frac{p+1}{p+n+2}$ of the probability that

[3] Ripper also supports bag-of-words representations by allowing conditions of the form *word* $\geq n$ to encode the fact that *word* has to occur at least n times. However, this option is hardly ever used in the literature, and we have not encountered a case where it significantly improved the results.

[4] We have also experimented with ordered rule sets, but the results were usually a little worse. Besides, in "ordered" mode, Ripper treats one class as the default class and does not learn rules for that class. We have not yet tried the round-robin approach, which learns one classifier for each pair of classes and seems to outperform both ordered and unordered rule sets [34].

an example covered by the rule is positive (estimated on the p positive and n negative instances that the rule covers on the training set). This is the same confidence estimate that Ripper uses internally for tie-breaking on unordered rule sets, when multiple rules fire and predict different classes for the same example. The only extension we make is that we assign a confidence score of 0 to all predictions for which none of the rule sets for the individual classes fired, i.e., for those prediction that originate from a default rule (i.e., a rule that simply predicts the majority class). Ripper adds such rules to the end of a theory in order to guarantee full coverage of the example space. The reasons for ignoring predictions from such default rules are that on the one hand we do not need full coverage (as the individual predictions are combined into an ensemble), and that on the other hand it turned out that many of the learned rules have a fairly low coverage, so that many of the examples can only be classified using a default rule. Thus, it may happen quite frequently that a few good rules are outnumbered by a large number of default predictions. Some of these preliminary results may be found in [32]. Note that an example may be covered by multiple rules with contradicting class predictions, so that more than one class may have a confidence score > 0.

We implemented the following voting schemes:

Voting *(Vote)*: The simplest technique is to give each member of the hyperlink ensemble one vote, and predict the class that receives the most votes. Ties are broken in favor of larger classes, and predictions with 0 confidence score are ignored.

Weighted Sum *(Weight)*: The *Weight* voting scheme is identical to the previous technique except that confidence scores are used to weight the vote of each ensemble member. Thus, a few predictions with high confidence scores may over-rule a larger number of predictions with lower scores.

Weighted Normalized Sum *(Norm)*: This voting scheme is identical to the previous one, except that the confidence scores are first normalized in a way that distributes a total weight of 1 among the different candidate classes for each link. This is necessary because the confidence score that is associated with each class only depends on the number of positive and negative examples covered by the best rule that predicts this class and covers the example. Therefore the confidence scores associated with each class cannot be interpreted as class probability estimates unless they are normalized.

Maximum Confidence *(Max)*: The last combination method simply chooses the class prediction that receives the highest confidence among all member of the ensemble and predicts that class. This is an attempt to use only the most accurate of all applicable rules to classify a page.

Of course, more elaborate prediction combiners are thinkable. Note, however, that the number of predictions that need to be combined varies from example to example (depending on the number of predecessor pages). Hence, the application of meta-learning approaches like stacking [66] or arbiters and combiners [14] is not straightforward.

Class	Links		Pages	
Student	2128	36.67%	544	51.81%
Faculty	1374	23.68%	147	14.00%
Course	915	15.77%	229	21.81%
Project	701	12.08%	83	7.90%
Department	530	9.13%	4	0.38%
Associate	121	2.09%	30	2.86%
Post Doc	34	0.59%	13	1.24%

Table 1: Class distribution of the data in links and pages.

5.3 The WEB→KB Dataset

The dataset we used for demonstrating the viability of our approach consists of 1050 pages that are part of a data set collected for the WEB→KB project [20].[5] The pages were manually classified into one of the categories *Student*, *Department*, *Faculty*, *Research Project*, *Research Associate*, *Post Doc*, and *Course*. Within these pages, 5803 hyperlinks point to another page within this set. Table 1 shows the class distribution of the dataset for the 1050 pages and for the 5803 hyperlinks. The classes have very different characteristics. First of all, they differ considerably in their sizes. Second, the average number of links that point to a page varies. For example, there are only four home pages of computer science departments in the entire set, which would be too few data to apply a straight-forward text classification algorithm. However, there are more than 100 links pointing to each of these pages, which makes *Department* a class that can be learned fairly well.

The pages/links were collected from four universities, Cornell (243/1073), Texas (253/1264), Washington (256/1771), and Wisconsin (298/1695). For constructing the hyperlink ensembles, we first discarded the entire text of each target page. Instead, we identified the set of predecessor pages that contain a pointer to the target page. This was done by scanning the dataset for all occurrences of an HREF that contains the address of the current page. In principle, this could also be performed on-line using a search engine like GOOGLE that allows to query for pages that point to a given address. From these predecessor pages, we identified the relevant parts of the documents (e.g., the anchor text of the hyperlink to the target page), removed all HTML markers and sentence marks from the text and converted all words to lower case thereby replacing special characters within a word with '_' and all digits with the letter 'D'.

5.4 Evaluation Procedure

All reported results are from a 4-fold leave-one-university-out cross-validation, i.e., for each experiment we combined the examples of three universities to learn a classifier which was then tested on the data of the fourth university. Because of the different test set sizes for each of the four results, we used micro-averaging for evaluating the accuracy of the predictors, i.e., we lumped all predictions from all four runs together and computed an accuracy measure on the entire set of predictions.

[5] A somewhat different version of this dataset is available from `http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/`.

	Accuracy
Hyperlink Ensembles (anchor text)	
Vote	74.67
Normal	74.38
Weight	74.19
Max	74.76
Full-text Classifier (set-of-words)	70.67
Feature Subset Selection	
(50%)	73.90
(10%)	74.19
(5%)	74.76
(1%)	71.33
(0.1%)	54.67

Table 2: Accuracy results for anchor-text-based hyperlink ensembles with a full-text classifier and a full-text classifier + feature selection.

6 Comparison to full-text classifier

Table 2 shows the results of hyperlink ensembles based on features using the anchor text of predecessor pages. The results of a full-text classifier, which uses the set of words occurring in the text of the target pages as a feature set, are shown below. The hyperlink ensembles outperform the full-text classifier by a significant margin, even though they only use the text that occurs on anchor texts that point to the target page and completely ignore the text on the page itself.

One reason for the better performance of the hyperlink ensemble could be that it uses considerably less features than the full-text classifier. There is considerable evidence that a reduction of the features will improve classification accuracy in text domains [43, 47, 67]. Could we obtain a similar accuracy by employing feature subset selection on the full-text classifier? To answer this we also compared the performance of hyperlink ensembles to a full-text classifier that uses feature subset selection as a pre-processor. Feature subset selection was performed using class entropy as implemented in the filter-text program that comes with Ripper. Table 2 shows the results for using only the top n% of the features of the full-text classifier (selected by entropy). As can be seen, at the best level (5% to 10%) feature subset selection performs as well as the hyperlink ensembles.

So we could conclude from these first experiments that—in this domain—hyperlink ensembles perform as well as an optimized full-text classifier, but without the need for the expensive feature subset selection process, and, in particular, without having to know (or experimentally determine) the right level of feature reduction. However, note that we have not optimized the hyperlink ensemble in any way. We have not checked whether feature subset selection would improve the link-classifiers as well, and, more importantly, we only focussed on the anchor texts of predecessor pages, which provide a fairly limited amount of information. In the next section, we will evaluate alternative ways of obtaining feature sets for the predecessor pages.

7 Alternative Features Sets

The working hypothesis behind our research is that the information on predecessor pages may often be more informative than the information on the page itself. However, we have not yet determined on what parts of the predecessor page the relevant information can be found. So far, we have only considered the anchor text of the link that points to the target page, which—in the experiments reported in the previous section—has already given a satisfactory performance. Nevertheless, if we take another look at Figure 1, the anchor text can only be helpful for classifying the left-most of the three predecessor pages. If this page was missing, our anchor-text-based hyperlink ensemble would not be able to determine that Dr. Marx is a professor, although the information is clearly present in the other two pages.

Using this picture as a motivation, we argue that in many cases at least one of the following three pieces of information provides an obvious clue for the intended classification of the page:

- the anchor text
- the context in which the anchor text appears
- the headings that structurally precede the section of the document in which the link occurs

Each of these three cases can handle one of the three predecessor pages shown in Figure 1, and we believe that they reflect general regularities, at least within the domain we studied.

For example, department pages typically have a large number of links pointing to them that are marked with anchor texts like “computer science department”, “CS department”, “dept. of computer science”, or similar. Each of them should be sufficient to identify the link as pointing to the page of a computer science department.

Student home pages very often contain a pointer to their advisor’s home page. Thus, faculty home pages can often be identified by the occurrence of the word “advisor” in the neighborhood of a link that points to the page. As student home pages are usually contain many links, it is quite likely that there are several links occurring near the word “advisor”. Thus it is reasonable to assume that the use of linguistic information could help to further focus the learner. For example, being able to recognize a sentence fragment like “my advisor is `<...>`” makes the learner’s task trivial.

However, not all pages contain grammatical text. This is particularly true for so-called hub pages, which are collections of pointers to authoritative information sources on a topic [39], or for pages that provide a collection of links to pages that are all instances of a certain group of objects. For example, many computer science departments have a page that lists all students, faculty, staff, projects, courses, or other information. Typically such pages (or segments of pages) start with a heading that identifies the type of information that is listed below (see, e.g., the right-most predecessor page in Figure 1). Clearly, this information can also be very useful for classifying the pages that are pointed to in the list below this heading.

In order to capture such regularities, we implemented four different feature extraction techniques:

Anchor: All words that occurred in the anchor text of the link (between the opening <A ...> and the closing </A> of the HTML link). This is the representation used in the experiments of the previous section.

Heading: All words that occurred in headings that *structurally* precede the hyperlink in the HTML document. This means a heading of type <Hi> is included iff it appears before the hyperlink and no heading of type <Hj> with $j \leq i$ appears in the segment between the heading and the hyperlink. Page titles and titles for definition lists (<DT>) were also included as headings (with $i = 0$ and $i = 7$ respectively). Throwing the words of all headings together is a crude approximation because headings further away could contain misleading information (e.g., department pages that are structured into sections with informations about various aspects of departments). Our motivation for using all headings, however, is that it is not always the heading immediately preceding the link that contains the relevant information. Future work might weight the headings differently according to their structural distance to the hyperlink.

Paragraph: All words of the paragraph in which the hyperlink occurs. Our method for determining the paragraph is somewhat heuristic: Pieces of text separated by <P> or an empty line are paragraphs, as are structural entities such as items in a list <LI>. Our straightforward method for grabbing the paragraphs is not perfect. E.g., it is hard to determine what to do with the break marker
, which is sometimes used as a simple line break, occasionally even within a sentence, but sometimes also for generating artificial list environments. As one of our goals for the paragraph grabber was to provide a reasonable text fragment for phrase extraction via AutoSlog (see below), we tried to resolve such issues conservatively.

Phrases: We used the text fragment extracted for the paragraph information to generate phrasal features of the sort described in [35]. AutoSlog-TS, originally conceived for information extraction [58, 59], is able to provide a learner with features that capture some of the syntactic structure of natural language text. We used a version of AutoSlog-TS that uses hyperlinks as a target[6] and returns patterns that capture the syntactic role of the prepositional phrase on the hyperlink. Consider for example the sentence *"My advisor is* <A HREF="http:/..."> *Tom Mitchell*</A>*."*. From the above input sentence, AutoSlog-TS would extract the feature *advisor_is_<_>* that would match all pages in which a student uses this phrase to refer to her advisor's home-page. Note that this type of feature is able to provide a better focus than simply using the word *advisor* in the paragraph feature, in particular in cases where more than one hyperlink appears in the paragraph.

[6]The main difference between the version used in [35] and the version we used in these experiments is that the former extracts *all* syntactic patterns from a text, while the latter specifically targets phrases around hyperlinks.

Classifier	Combination Method			
	Vote	Normal	Weight	Max
Default	51.81	51.81	51.81	51.81
Anchor	74.67	74.38	74.19	**74.76**
Headings	72.29	72.38	**72.95**	**72.95**
Paragraph	66.86	66.86	**66.95**	66.29
Phrases	56.29	56.29	56.38	**56.57**
Anchor+Headings	85.33	84.95	85.14	**86.57**
Anchor+Paragraph	74.29	74.00	73.90	**74.67**
Anchor+Phrases	74.95	74.76	74.48	**75.43**
Headings+Paragraph	79.90	80.19	81.14	**81.33**
Headings+Phrases	**77.33**	76.86	76.95	77.14
Paragraph+Phrases	66.86	**67.05**	66.86	66.10
w/o Anchor	76.57	76.76	76.48	**77.05**
w/o Headings	73.90	74.10	74.19	**74.38**
w/o Paragraph	85.52	84.86	85.43	**86.86**
w/o Phrases	82.29	81.71	82.67	**83.24**
All	84.54	84.57	**85.05**	**85.05**

Table 3: Accuracies for classifying the 1050 pages using various methods for combining link predictors to page predictors.

We evaluated each of the above feature representations in isolation, but also looked at all possible combinations of these feature sets, i.e., we first ran experiments using only single feature sets, then all pairs of feature sets, all triplets, and finally using all four feature sets. If more than one feature set was used, each one was encoded as a separate set-valued feature, i.e., identical words in different feature sets were treated as different features. We will refer to the set-valued feature as a *feature set* and use the terms *feature* and *word* (or *phrase* for the *Phrases* feature set) interchangably.

7.1 Accuracy

Table 3 shows the accuracies measured for predicting the page labels. The rows list the different representation schemes, starting from the default prediction accuracy (using no features) to the classifier that uses all features. The columns of the table give the accuracy for each of the four implemented prediction combination techniques voting, normalized weighted average, weighted average, and finally the maximum method (see Section 5.2).

The first thing to notice is that some results are considerably better than those reported in Table 2. For example, if all feature sets are used, all results are near the 85% mark, which is more than 10% above the best result obtained by the full-text classifier. This is clear evidence that hyperlink ensembles, when used with a sufficiently rich representation of the predecessor pages, can significantly outperform full-text classifiers. It should also be noted that this classifer, which uses all four features sets, uses less

than half of the features of the full-text classifier (8,075 vs. 20,322).[7]

The results also show that adding a new feature set will in general increase the obtained predictive accuracy. The exception to the rule is the *Paragraph* feature set. Whenever its features are added to a representation that already includes the *Anchor* features, the result is a loss of predictive accuracy. A reason for this might be that these two feature sets are much less independent of each other than other pairs of feature sets.[8] The features in the *Phrases* feature set are used quite infrequently in the learned rules, and in general do not make a big difference. It should nevertheless be noted that the highest accuracy (86.86%) was achieved when using all but the *Paragraph* feature set (and combining the predictions of the links by chosing the one with maximum confidence) although the differences between this version and the version that uses only *Anchor* and *Headings* are inconclusive.

Among the four different techniques for combining the link predictions to a page prediction, taking the prediction with the maximum confidence is a clear winner. In 12 out of the 15 runs, using this method gave the best results (shown in **bold** face). It is interesting to note that this method, which focuses on the example that is considered to be most relevant and ignores all others, loosely corresponds to the so-called *single-tuple bias* in multi-instance learning [15] (see also footnote 2). However, in general, the differences between the combination methods are not nearly as large as the differences between the different document representations.

7.2 Recall and Precision

We have discussed above that many of the test examples are classified using the default rule and that it seems to be advisable to ignore these default link predictions for computing the page predictions. But what happens in cases where *all* links that point to a page are classified by default rules, i.e., no link contains any information that could be used for a justified prediction? In the experiments reported in the previous section, we have simply predicted the majority class *Student* for each of these pages. What if we ignore these predictions? It can be expected that the classification accuracy goes up at the expense of classifying fewer pages. This trade-off is commonly measured in terms of precision and recall.

Table 4 lists recall and precision estimates that shed some light upon this question. *Recall* is the percentage of pages which were classified using at least one rule other than the default rule. Note that this estimate is the same for all combination methods because the underlying link classifiers are the same and hence the links that are classified with default rules are the same. *Precision* is the percentage of correct predictions among all non-default predictions, i.e., it measures the prediction accuracy for only those cases where a learned rule was used. In general, the precision scores are much higher than

[7] The reported number of features are the total number of features in the entire dataset. Each of the four training sets (which used only part of the data) contained on average a little more than 80% of the features for both types of classifiers.

[8] Note, however, that even though the set of words occurring on the anchor text is a subset of the set of words occurring in its surrounding paragraph, the resulting *Anchor* feature set is *not* a subset of the resulting *Paragraph* feature set because the feature x_`occurs_in_anchor_text` is semantically different from the feature x_`occurs_in_paragraph`, the former being more specific than the latter. This is a result of the encoding as separate set-valued features.

Classifier	Recall	Precision			
		Vote	Normal	Weight	Max
Anchor	40.76	83.64	82.30	82.48	83.88
Headings	74.10	88.05	88.19	88.95	88.95
Paragraph	46.67	75.71	75.91	75.92	74.49
Phrases	10.57	70.27	66.94	71.17	72.97
Anchor+Headings	85.90	92.35	91.91	92.13	93.79
Anchor+Paragraph	60.19	83.23	82.81	82.59	83.86
Anchor+Phrases	42.38	82.92	81.68	81.80	84.04
Headings+Paragraph	82.38	88.44	88.79	89.94	90.17
Headings+Phrases	74.48	92.20	91.58	91.69	91.94
Paragraph+Phrases	46.95	78.09	78.43	78.09	76.47
w/o Anchor	65.71	86.67	86.98	86.52	87.39
w/o Headings	57.71	83.17	83.39	83.66	83.99
w/o Paragraph	86.00	92.14	91.36	92.03	93.69
w/o Phrases	78.00	86.94	86.20	87.42	88.16
All	87.05	91.03	91.14	91.68	91.68

Table 4: Recall and Precision for the page predictors. *Recall* is the percentage of pages that are not classified with the default rule and *Precision* is how many of these classifications were correct.

the accuracy results of Table 3. This is not surprising because accuracy can be viewed as a weighted sum between the precision on the recalled examples and the precision on the examples classified by default rules (which should be about the default accuracy, although the variance can be very high).

More informative are the recall results. For example, phrasal features only have a recall of slightly more than 10%. This means that only very few pages are classified with a non-default rule if only this representation is used. Adding this feature set to other features usually does not increase recall although small increases in accuracy and precision are noticable. This is consistent with previous work where it was shown that such features can increase precision at low levels of recall [35]. The low recall of *Paragraph* is somewhat surprising, as one could expect that the high number of features should be able to achieve a higher coverage. However, a higher number of features also increases the danger of overfitting, which, in our opinion, is what happens with the *Paragraph* feature sets. *Headings* features have by far the highest recall. We explain this with the fact that the words occuring in headings tend to be more repetitious.

8 Effectiveness of Hyperlink Ensembles

One question that remains unanswered by Table 3 is how much do we gain by combining the predictions of different links pointing to the same page, i.e., how good are the ensembles in exploiting the redundancy provided by multiple classifications within the ensemble. If, for example, all predictions within the ensemble uniformly predict

	Combination Method				
Encoding	No	Vote	Normal	Weight	Max
Default	36.67	36.67	36.67	36.67	36.67
Anchor	57.92	75.93	75.56	75.37	**76.05**
Headings	43.34	66.62	69.89	**70.77**	64.33
Paragraph	53.40	65.91	65.81	**66.33**	58.59
Phrases	39.12	56.38	56.38	56.47	**56.83**
Anchor+Headings	62.49	86.18	85.46	**86.25**	83.22
Anchor+Paragraph	58.40	**73.70**	73.67	73.46	71.81
Anchor+Phrases	58.43	74.39	74.10	73.79	**75.53**
Headings+Paragraph	58.50	78.67	78.98	**80.30**	76.63
Headings+Phrases	45.03	76.77	75.50	76.10	**79.63**
Paragraph+Phrases	51.84	63.52	**63.76**	63.45	60.18
w/o Anchor	55.56	78.01	**78.25**	75.87	76.55
w/o Headings	55.85	71.19	**71.62**	69.95	65.95
w/o Paragraph	59.80	**83.68**	82.22	83.65	82.65
w/o Phrases	57.99	79.15	77.74	**79.44**	79.20
All	63.11	82.35	82.68	**83.68**	79.27

Table 5: Accuracies for classifying the 5803 links with various predictor combination methods. Except for the column labelled *No*, the prediction in each column were generated by assigning to each link the classification of the page it points to, which was derived as in table 3.

the same class label, there would be no need for combining the predictions; we could simply use any of the predictions for the individual hyperlinks.

To investigate this question, we computed a weighted accuracy estimate by weighting each page with the number of links that point to that page. In other words, all links that point to the same page perform an internal vote to decide upon a common classification for the page they point to. Each link of such a group will then predict this common label. The resulting accuracy estimate counts the number of correctly predicted links over all links, and can thus be directly compared to the accuracies of the base classifiers that predict the class labels of each link independently.

These results are shown in Table 5. The first thing to notice is a substantial difference between the independent classifier (first column) and the classfiers that rely on combining the predictions for different links. Obviously, many mistakes could be corrected by combining the predictions of different links. It is also interesting to observe that the *Max* prediction method (last column) is not as dominant as in Table 3 and, in some cases, it performs substantially worse than its competitors. The reason for this is that the maximum prediction is much less susceptible to variations in the *number* of link predictions that are combined to a single page prediction. If an erroneous link prediction has the maximum confidence score, it is used for predicting the class of the page. The voting and weighting methods, on the other hand, can make use of a number of unanimous predictions with lower confidence scores to override a prediction with a higher confidence score. Thus, it can be expected that pages with a higher number of

incoming links are classified more reliably by voting or weighting, while pages with a lower number incoming links are better classified by taking the prediction with the maximum score. As the latter category is more frequent, the accuracies tend to be higher for the maximum prediction method in Table 3, while they tend to be higher for the voting and weighting schemes in Table 5.

In summary, these results illustrate that the use of hyperlink ensembles allows to effectively correct erroneous predictions of the base classifier of the ensemble.

9 Discussion

In this paper, we have investigated the idea of forming an ensemble classifier for hypertext documents by combining information from pages that have a hyperlink pointing to the page. This approach allows to exploit a richer vocabulary, independent assessments of page contents through multiple authors, and redundancy in classification. The fact that we do not depend on the text on the target page provides us a with a simple mechamism for dealing with pages that have sparse text, no text, or text in different languages.

We implemented the technique using a rule-base classifier, which was trained to predict the class labels of individual links. The predictions of links pointing to the same page were then combined in four different ways to yield predictions for this page. The best results were achieved by selecting the prediction with the maximum confidence from the ensemble. Among four different ways for identifying relevant information on the predecessor pages—the anchor text, the headings that structurally precede it, the text of the paragraph in which it occurs, and a set of linguistic phrases the capture the syntactic role of the anchor text in this paragraph—headings and anchor text seem to be most useful. Linguistic phrases seem to provide a better focus using the raw text of the paragraphs, but the they do not seem to be as useful as in previous work [35], the main problem being their low coverage.

Although the encoding schemes we used are rather simple, our experiments illustrate that the use of information about the HTML structure of pages and about the structure of the Web itself can be useful for improving text categorization on the Web. The use of more elaborate representation schemes (e.g., including information about the type of headings or even using an entire HTML-tree as in [29]) suggests itself as a rewarding topic for further research, as does the use of relational learning techniques, such as those introduced in [19, 21], which do not exploit the redundancy provided by multiple incoming links but, instead, are able to use hyperlink paths of arbitrary length. In addition, we also think of combining full-text classifiers with hyperlink ensembles (e.g., by simply including the full-text classifier into the ensemble), and of having separate classifiers for each feature sets (thereby integrating our hyperlink ensembles with ensemble techniques that are based on multiple feature sets [e.g., 4]).

In a way, our hyperlink ensembles are similar to some previous results which showed that the use of class information from predecessor (and successor) pages may improve classification accuracy [12, 50, 68]. The main difference is that these techniques use the predicted *classes* of the predecessor pages as intermediate results for the prediction of the target page, while we propose to predict the category of the tar-

get page using *features* derived from its predecessor pages. As described in Section 2, the above-mentioned techniques also have to rely on some iterative technique because the class labels of neighboring pages depend on each other. Furthermore, it makes the strong assumption that all neighboring pages also have some class label attached to it. Our technique does not make such an assumption, which we believe does not hold for the open domain of Web pages.

The main limitation of our work is that we only evaluated our technique on a single domain, for which we obtained very good results (the base-line performance of a full-text classifier optimized with feature subset selection was improved by more than 10%). It should be noted that our experiments were performed off-line on this dataset, which has certain peculiarities that might make it exceptionally well-suited for this kind of approach. For example, all the pages were collected using a Web crawler which retrieved all pages that could be reached by some path from the top-level pages of four computer science departments without leaving the domain of these departments. This procedure guaranteed that all pages originated from a fairly narrow domain (computer science in academia). It remains an open question, how our techniques would perform in an open domain, i.e., for classifying arbitrary pages with arbitrary hyperlinks, but we believe that our techniques are well-suited to this type of problem because—contrary to previous approaches—we do not make the assumption that all (or some) predecessor pages can be assigned to meaningful classes, and we are able to exploit the redundancy provided by multiple links pointing to a page.

Acknowledgements

The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry of Education, Science and Culture. Parts of this work were performed during the author's stay at Carnegie Mellon University, which was enabled by a *Schrödinger-Stipendium* (J1443-INF) of the Austrian *Fonds zur Förderung der wissenschaftlichen Forschung (FWF)*. The author has benefited greatly from discussions with Mark Craven, Tom Mitchell, Ellen Riloff, and the members of the CMU text learning group. Thanks go also to Ellen Riloff for providing AutoSlog-TS and support in using it.

References

[1] Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2000.

[2] Robert Armstrong, Dayne Freitag, Thorsten Joachims, and Tom Mitchell. WebWatcher: A learning apprentice for the world wide web. In C. Knoblock and A. Levy, editors, *Proceedings of AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments*, pages 6–12. AAAI Press, 1995. Technical Report SS-95-08.

[3] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36:105–169, 1999.

[4] Stephen D. Bay. Nearest neighbor classification from multiple feature subsets. *Intelligent Data Analysis*, 3(3):191–209, 1999.

[5] Adam Berger. Error-correcting output coding for text classification. In *Proceedings of the IJCAI-99 Workshop on Machine Learning for Information Filtering*, 1999.

[6] Krishna Bharat, Andrei Broder, Monika R. Henzinger, Puneet Kumar, and Suresh Venkatasubramanian. The connectivity server: Fast access to linkage information on the Web. *Computer Networks*, 30(1–7):469–477, 1998. Proceedings of the 7th International World Wide Web Conference (WWW-7), Brisbane, Australia.

[7] Krishna Bharat and Monika R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Proceedings of the 21st ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-98)*, pages 104–111, 1998.

[8] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[9] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks*, 30(1–7):107–117, 1998. Proceedings of the 7th International World Wide Web Conference (WWW-7), Brisbane, Australia.

[10] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph structure in the Web. In Ivan Herman and Albert Vezza, editors, *Proceedings of the 9th International World Wide Web Conference (WWW-9)*, Amsterdam, The Netherlands, 2000. Foretec Seminars, Inc.

[11] Soumen Chakrabarti. Data mining for hypertext: A tutorial survey. *SIGKDD explorations*, 1(2):1–11, January 2000.

[12] Soumen Chakrabarti, Byron Dom, and Piotr Indyk. Enhanced hypertext categorization using hyperlinks. In *Proceedings of the ACM SIGMOD International Conference on Management on Data*, pages 307–318, Seattle, WA, 1998. ACM Press.

[13] Soumen Chakrabarti, Byron Dom, Prabhakar Raghavan, Sridhar Rajagopalan, David Gibson, and Jon Kleinberg. Automatic resource compilation by analyzing hyperlink structure and associated text. *Computer Networks*, 30(1–7):65–74, 1998. Proceedings of the 7th International World Wide Web Conference (WWW-7), Brisbane, Australia.

[14] Philip K. Chan and Salvatore J. Stolfo. A comparative evaluation of voting and meta-learning on partitioned data. In *Proceedings of the 12th International Conference on Machine Learning (ICML-95)*, pages 90–98. Morgan Kaufmann, 1995.

[15] Yann Chevaleyre and Jean-Daniel Zucker. A framework for learning rules from multiple instance data. In Luc De Raedt and Peter Flach, editors, *Proceedings of the 12th European Conference on Machine Learning (ECML-01)*, pages 49–60, Freiburg, Germany, 2001. Springer-Verlag.

[16] William W. Cohen. Fast effective rule induction. In A. Prieditis and S. Russell, editors, *Proceedings of the 12th International Conference on Machine Learning (ML-95)*, pages 115–123, Lake Tahoe, CA, 1995. Morgan Kaufmann.

[17] William W. Cohen. Learning rules that classify e-mail. In M.A. Hearst and H. Hirsh, editors, *Proceedings of the AAAI Spring Symposium on Machine Learning in Information Access*, pages 18–25. AAAI Press, 1996. Technical Report SS-96-05.

[18] William W. Cohen. Learning trees and rules with set-valued features. In *Proceedings of the 13th National Conference on Artificial Intelligene (AAAI-96)*, pages 709–716. AAAI Press, 1996.

[19] Mark Craven. Using statistical and relational methods to characterize hyperlink paths. In D. Jensen and H. Goldberg, editors, *Artificial Intelligence and Link Analysis: Papers from the 1998 AAAI Fall Symposium*, pages 14–20, Orlando, Florida, 1998. AAAI Press. Technical Report FS-98-01.

[20] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom Mitchell, Kamal Nigam, and Seán Slattery. Learning to construct knowledge bases from the World Wide Web. *Artificial Intelligence*, 118(1-2):69–114, 2000.

[21] Mark Craven and Seán Slattery. Relational learning with statistical predicate invention: Better models for hypertext. *Machine Learning*, 43(1-2):97–119, 2001.

[22] Mark Craven, Seán Slattery, and Kamal Nigam. First-order learning for Web mining. In C. Nédellec and C. Rouveirol, editors, *Proceedings of the 10th European Conference on Machine Learning (ECML-98)*, pages 250–255, Chemnitz, Germany, 1998. Springer-Verlag.

[23] Jeffrey Dean and Monika R. Henzinger. Finding related pages in the World Wide Web. In A. Mendelzon, editor, *Proceedings of the 8th International World Wide Web Conference (WWW-8)*, pages 389–401, Toronto, Canada, 1999.

[24] Thomas G. Dietterich. Machine learning research: Four current directions. *AI Magazine*, 18(4):97–136, Winter 1997.

[25] Thomas G. Dietterich. Ensemble methods in machine learning. In J. Kittler and F. Roli, editors, *First International Workshop on Multiple Classifier Systems*, pages 1–15. Springer-Verlag, 2000.

[26] Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–158, 2000.

[27] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2: 263–286, 1995.

[28] Thomas G. Dietterich, Richard H. Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89 (1–2):31–71, 1997.

[29] Dan DiPasquo. Using HTML structure to aid in automatic information retrieval from the world wide web. Senior Honors Thesis, School of Computer Science, Carnegie Mellon University, May 1998.

[30] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[31] Johannes Fürnkranz. Pruning algorithms for rule learning. *Machine Learning*, 27 (2):139–171, 1997.

[32] Johannes Fürnkranz. Exploiting structural information for text classification on the WWW. In D.J. Hand, J. N. Kok, and M.R. Berthold, editors, *Advances in Intelligent Data Analysis: Proceedings of the 3rd International Symposium (IDA-99)*, pages 487–489, Amsterdam, Netherlands, 1999. Springer-Verlag.

[33] Johannes Fürnkranz. Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1):3–54, February 1999.

[34] Johannes Fürnkranz. Round robin rule learning. In C. E. Brodley and A. P. Danyluk, editors, *Proceedings of the 18th International Conference on Machine Learning (ICML-01)*, pages 146–153, Williamstown, MA, 2001. Morgan Kaufmann Publishers.

[35] Johannes Fürnkranz, Tom Mitchell, and Ellen Riloff. A case study in using linguistic phrases for text categorization on the WWW. In M. Sahami, editor, *Learning for Text Categorization: Proceedings of the 1998 AAAI/ICML Workshop*, pages 5–12, Madison, WI, 1998. AAAI Press. Technical Report WS-98-05.

[36] Johannes Fürnkranz and Gerhard Widmer. Incremental Reduced Error Pruning. In W. Cohen and H. Hirsh, editors, *Proceedings of the 11th International Conference on Machine Learning (ML-94)*, pages 70–77, New Brunswick, NJ, 1994. Morgan Kaufmann.

[37] Rayid Ghani. Using error-correcting codes for text classification. In P. Langley, editor, *Proceedings of the 17th International Conference on Machine Learning (ICML-00)*, pages 303–310, Stanford, CA, 2000. Morgan Kaufmann.

[38] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In C. Nédellec and C. Rouveirol, editors, *Proceedings of 10th European Conference on Machine Learning (ECML-98)*, pages 137–142, Chemnitz, Germany, 1998. Springer-Verlag.

[39] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, September 1999. ISSN 0004-5411.

[40] John F. Kolen and Jordan B. Pollack. Back propagation is sensitive to initial conditions. In *Advances in Neural Information Processing Systems 3 (NIPS-90)*, pages 860–867. Morgan Kaufmann, 1991.

[41] Ken Lang. NewsWeeder: Learning to filter netnews. In A. Prieditis and S. Russell, editors, *Proceedings of the 12th International Conference on Machine Learning (ML-95)*, pages 331–339. Morgan Kaufmann, 1995.

[42] Leah S. Larkey and W. Bruce Croft. Combining classifiers in text categorization. In Hans-Peter Frei, Donna Harman, Peter Schäuble, and Ross Wilkinson, editors, *Proceedings of the 19th ACM International Conference on Research and Development in Information Retrieval (SIGIR-96)*, pages 289–297, Zürich, CH, 1996. ACM Press, New York, US.

[43] David D. Lewis. Feature selection and feature extraction for text categorization. In *Proceedings of a Workshop on Speech and Natural Language*, pages 212–217, Harriman, NY, 1992.

[44] Henry Lieberman. Letizia: An agent that assists web browsing. In C. S. Mellish, editor, *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 924–929. Morgan Kaufmann, 1995.

[45] Eddy Mayoraz and Miguel Moreira. On the decomposition of polychotomies into dichotomies. In *Proceedings of the 14th International Conference on Machine Learning (ICML-97)*, pages 219–226, Nashville, TN, 1997. Morgan Kaufmann.

[46] Oliver A. McBryan. GENVL and WWWW: Tools for taming the Web. In *Proceedings of the 1st World-Wide Web Conference (WWW-1)*, pages 58–67, Geneva, Switzerland, 1994. Elsevier.

[47] Dunja Mladenić. Feature subset selection in text-learning. In C. Nédellec and C. Rouveirol, editors, *Proceedings of the 10th European Conference on Machine Learning (ECML-98)*, pages 95–100, Chemnitz, Germany, 1998. Springer-Verlag.

[48] Dunja Mladenić. Turning Yahoo into an automatic web-page classifier. In H. Prade, editor, *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98)*, pages 473–474, Brighton, U.K., 1998. Wiley.

[49] Dunja Mladenić. Text-learning and related intelligent agents: A survey. *IEEE Intelligent Systems*, 14(4):44–54, July/August 1999.

[50] Hyo-Jung Oh, Sung Hyon Myaeng, and Mann-Ho Lee. A practical hypertext categorization method using links and incrementally available class information. In *Proceedings of the 23rd ACM International Conference on Research and Development in Information Retrieval (SIGIR-00)*, pages 264–271, Athens, Greece, 2000.

[51] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.

[52] Terry R. Payne and Peter Edwards. Interface agents that learn: An investigation of learning issues in a mail agent interface. *Applied Artificial Intelligence*, 11(1): 1–32, 1997.

[53] Michael Pazzani, Jack Muramatsu, and Daniel Billsus. Syskill & Webert: Identifying interesting web sites. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI-96)*, pages 54–61. AAAI Press, 1996.

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

[55] J. Ross Quinlan. Determinate literals in inductive logic programming. In *Proceedings of the 8th International Workshop on Machine Learning (ML-91)*, pages 442–446, 1991.

[56] Jason D. M. Rennie. ifile: An application of machine learning to e-mail filtering. In M. Grobelnik, D. Mladenić, and N. Milic-Frayling, editors, *Proceedings KDD-2000 Workshop on Text Mining*, Boston, MA, 2000.

[57] Bradley L. Richards and Raymond J. Mooney. Learning relations by pathfinding. In *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI-92)*, pages 50–55, San Jose, CA, 1992. AAAI Press.

[58] Ellen Riloff. Automatically generating extraction patterns from untagged text. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI-96)*, pages 1044–1049. AAAI Press, 1996.

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

[60] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.

[61] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[62] Alexander K. Seewald and Johannes Fürnkranz. An evaluation of grading classifiers. In *Advances in Intelligent Data Analysis: Proceedings of the 4th International Symposium (IDA-01)*, pages 115–124, Lisbon, Portugal, 2001. Springer-Verlag.

[63] Seán Slattery and Tom Mitchell. Discovering test set regularities in relational domains. In P. Langley, editor, *Proceedings of the 17th International Conference on Machine Learning (ICML-00)*, pages 895–902, Stanford, CA, 2000. Morgan Kaufmann.

[64] Ellen Spertus. ParaSite: Mining structural information on the Web. *Computer Networks and ISDN Systems*, 29(8-13):1205–1215, September 1997. Proceedings of the 6th International World Wide Web Conference (WWW-6).

[65] Sholom M. Weiss, Chidanand Apté, Fred J. Damerau, David E. Johnson, Frank J. Oles, Thilo Goetz, and Thomas Hampp. Maximizing text-mining performance. *IEEE Intelligent Systems*, 14(4):63–69, July/August 1999.

[66] David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–260, 1992.

[67] Yiming Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In D.H. Fisher, editor, *Proceedings of the 14th International Conference on Machine Learning (ICML-97)*, pages 412–420. Morgan Kaufmann, 1997.

[68] Yiming Yang, Seán Slattery, and Rayid Ghani. A study of approaches to hypertext categorization. *Journal of Intelligent Information Systems*, in press. Special Issue on Automatic Text Categorization.