Learning to Crawl: Comparing Classification Schemes

GAUTAM PANT
The University of Utah
and
PADMINI SRINIVASAN
The University of Iowa

Topical crawling is a young and creative area of research that holds the promise of benefiting from several sophisticated data mining techniques. The use of classification algorithms to guide topical crawlers has been sporadically suggested in the literature. No systematic study, however, has been done on their relative merits. Using the lessons learned from our previous crawler evaluation studies, we experiment with multiple versions of different classification schemes. The crawling process is modeled as a parallel best-first search over a graph defined by the Web. The classifiers provide heuristics to the crawler thus biasing it towards certain portions of the Web graph. Our results show that Naive Bayes is a weak choice for guiding a topical crawler when compared with Support Vector Machine or Neural Network. Further, the weak performance of Naive Bayes can be partly explained by extreme skewness of posterior probabilities generated by it. We also observe that despite similar performances, different topical crawlers cover subspaces on the Web with low overlap.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Search process*; I.2.6 [**Artificial Intelligence**]: Learning; I.5.4 [**Pattern Recognition**]: Applications; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Performance evaluation (efficiency and effectiveness)*

General Terms: Experimentation, Performance, Algorithms, Design

Additional Key Words and Phrases: Topical crawlers, focused crawlers, classifiers, machine learning

1. INTRODUCTION

Web crawler programs exploit the graph structure of the Web by starting at a *seed* page and then following the hyperlinks within it to attend to other pages. This process repeats with the new pages offering more hyperlinks to follow,

Authors' addresses: G. Pant, School of Accounting and Information Systems, The University of Utah, Salt Lake City, UT 84112; email: gautam.pant@business.utah.edu; P. Srinivasan, School of Library and Information Science and Department of Management Sciences; The University of Iowa, Iowa City, IA 52242; email: padmini-srinivasan@uiowa.edu.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 1515 Broadway, New York, NY 10036 USA, fax: +1 (212) 869-0481, or permissions@acm.org.
© 2005 ACM 1046-8188/05/1000-0430 $5.00

until a sufficient number of pages are fetched or some higher level objective is reached. With *topical* Web crawlers, the goal is to be selective about the pages fetched and ensure as best as possible that these are relevant to some initiating topic. Such crawlers, sometimes referred to as *focused* or *thematic* crawlers, are important for a variety of applications such as vertical search engines [McCallum et al. 2000], competitive intelligence [Chen et al. 2002; Pant and Menczer 2003], software agents [Chen et al. 1998; Menczer and Belew 2000; Chau et al. 2001; Pant and Menczer 2002], and digital libraries [Pant et al. 2004b; Qin et al. 2004]. The need for topical crawlers is fueled by the large size and the dynamic nature of the Web [Lawrence and Giles 1998]. While the widespread usage of such crawlers by millions of end users is not practical, we expect topical crawlers to be increasingly used for generating focused collections for research, preserving, and gathering intelligence (business or otherwise). For example, a business or a nonprofit organization may be interested in preserving temporal snapshots of cancer related information that appears on the Web over several years [Day 2003].

The significant challenge of identifying all and just the relevant or topical subspaces of the Web is typically addressed by building *intelligence* into the crawling strategy. This is generally achieved through appropriate heuristics to bias the search. The key decision emphasized in these heuristics is: Which is the next best hyperlink (URL) to follow given the crawler's current position in the Web as defined by the pages it has seen thus far? Examples of such heuristics include computing content similarity between the *parent* page that contains a hyperlink with a topic profile or query, measuring the centrality of the parent page based on the neighboring *Web graph*, and using *supervised* [Chakrabarti et al. 1999; McCallum et al. 2000] or *unsupervised* [Menczer and Belew 2000] machine learning to estimate the benefit of following a hyperlink.

A family of potential solutions for designing crawler heuristics comes from the area of classifiers. These machine learning tools, built using training sets of examples, have been studied extensively for a variety of purposes including the closely related task of text classification [McCallum and Nigam 1998; Dumais 1998; Joachims 2002]. In the context of topical crawler logic, classifiers offer a natural approach to decide if a given hyperlink is likely or is not likely to lead to a relevant Web page. It may be noted that the task in *classifier-guided* topical crawling is one of classifying URLs before downloading the pages corresponding to them. The current literature provides a few examples of research exploring classifiers for crawling. The highly cited 1999 paper by Chakrabarti et al. [1999] explores Naive Bayes classifiers in what the authors call a *focused* crawling framework. Similarly Diligenti et al. [2000] and Johnson et al. [2003] use Naive Bayes and Support Vector Machine (SVM) classifiers respectively to guide their topical crawlers. However, the research exploring the influence of classifiers on crawlers is limited to a small set of papers. Also, most of these explore a single classification scheme, which typically is Naive Bayes. Additionally, almost all of these investigations are limited to crawls for not more than 10 topics. These studies, although significant in that they draw our attention to the potential of classifiers for crawling, lead us to conclude that a systematic study exploring

alternate classification schemes under rigorous experimental conditions is now timely.

Our goal is thus to contribute such a systematic study, with experiments exploring multiple versions of the Naive Bayes, Support Vector Machine (SVM), and Neural Network classification schemes for crawling. These classification techniques are popular and well established in the areas of text and data mining with readily available implementations in several programming languages. We also study classifiers under varying conditions as for instance when there are just a few examples of relevant pages available to train the classifier. We perform our experiments using a collection of more than 100 topics, which allows us to make statistically valid conclusions. We have designed and developed a crawling framework that allows for flexible addition of new classifiers. The crawlers themselves are implemented as multithreaded objects that run concurrently. Our results show that Naive Bayes is a weak choice for guiding a topical crawler when compared with Support Vector Machine or Neural Network. Our post hoc analysis explains the better performance of some crawlers while recognizing the divergence among similar performing crawlers in terms of the subspaces of the Web covered by them.

The rest of this article is organized as follows. We begin by detailing related research in Section 2. Section 3 describes the underlying architecture of our crawlers. The various classification schemes and their versions used in this study are described in Section 4. The process of deriving topics for the evaluation of crawlers and the performance metrics are explained in Section 5 and Section 6 respectively. We present our experiments in Section 7 and illustrate the results in Section 8. Several observations, based on post hoc analysis, are made in Section 9 and Section 10 concludes our findings and enumerates future research directions.

2. RELATED RESEARCH

Starting from the early topical crawlers such as Fish Search [De Bra and Post 1994], a variety of methods have been proposed for building topical crawlers. Although crawlers based on machine learning algorithms are our focus, it is important to recognize the underlying research context exploring crawler design. Note that many crawler algorithms are variations of the best-first crawler, where a list of *unvisited* URLs is maintained as a priority queue. The unvisited URLs correspond to the pages that have not yet been downloaded by the crawler. The list of such URLs maintained by the crawler is called a *frontier*. Each unvisited URL has an associated score that reflects the benefit of following that URL and hence determines its priority for download. Variations of the best-first crawler may be produced by varying the heuristics used to score the unvisited URLs in the frontier. The heuristics affect the scoring of unvisited URLs and hence the path of the crawler. In a multi-threaded implementation [Pant et al. 2004a] the best-first crawler acts like a best-N-first crawler where N is a function of the number of simultaneously running threads. Thus best-N-first is a generalized version of the best-first crawler that picks N best URLs to crawl at a time.

The naive best-first crawler [Menczer et al. 2001; Pant et al. 2004a] measures the cosine similarity [Salton and McGill 1983] of a page to a topic profile or a query, and uses this similarity to estimate the benefit of following the hyperlinks found on that page. The naive best-first crawlers and their variations have been detailed and evaluated by us in multiple studies [Menczer et al. 2001; Pant et al. 2002; Menczer et al. 2004]. We found that more explorative (less greedy) best-N-first versions outperform a strict best-first crawler [Pant et al. 2002; Menczer et al. 2004]. FishSearch [De Bra and Post 1994], one of the earliest topical crawling algorithms uses a similarity measure like the one used in the naive best-first crawler for estimating the relevance of an unvisited URL. FishSearch also has a notion of depth-bound that prevents it from exploring paths that lead to a sequence of irrelevant pages. SharkSearch [Maarek et al. 1997; Hersovici et al. 1998; Ben-Shaul et al. 1999], while adopting the concepts of FishSearch, has a more refined notion of scores for unvisited URLs. The anchor text, text surrounding the links or *link context*, and scores inherited from ancestors, influence the scores of the URLs. The ancestors of a URL are the pages that appeared on the crawl path to the URL. SharkSearch, like its predecessor FishSearch, maintains a depth bound. That is, if the crawler finds unimportant pages on a crawl path up to a certain depth or distance, it stops crawling further along that path.

In general topical crawlers have used three different forms of contextual information to estimate the benefit of following a URL. These are:

Link context: This is the lexical content that appears around the given URL in the parent page. The parent page is the page from which the URL was extracted. The link context could range from the entire parent page or a select subset of it. Link context is utilized by almost all of the topical crawlers in one form or the other [De Bra and Post 1994; Hersovici et al. 1998; Chakrabarti et al. 1999; Menczer and Belew 2000; Diligenti et al. 2000; Aggarwal et al. 2001; Johnson et al. 2003; Pant et al. 2004b; Srinivasan et al. 2005].

Ancestor pages: The lexical content of pages leading to the current page that contains the given URL. A few topical crawlers such as those suggested by Diligenti et al. [2000] and Johnson et al. [2003] have attempted to use ancestor pages (in addition to the parent page). However, Johnson et al. [2003] have shown that the content of ancestor pages beyond the parent page provides very little help for topical crawling when compared with the parent page.

Web graph: The structure of the Web subgraph around the node (page) corresponding to the given URL may also provide essential cues about the merit of following the URL. For example, if the parent page in which the URL appears is a "good" hub,[1] then the priority for following the URL may be increased. This strategy was first utilized by Chakrabarti et al. [1999] in their focused crawler; however, no result (other than anecdotal comments) that pointed to its benefit was shown in that work. In a more recent investigation we have shown that exploitation of hubs can indeed produce statistically significant improvement in a topical crawler's performance [Pant and Menczer 2003].

[1]A page that has links to many resourceful pages.

In this article we focus on the use of the parent page as the context of the unvisited URLs within it. This commonly used contextual information is the most fundamental in classifier-based topical crawlers. Example Web pages are used as training instances and the trained classifier is used to judge the crawled pages in order to prioritize the unvisited URLs extracted from them. The use of the parent page to score the hyperlinks in it is founded on the topical locality hypothesis [Davison 2000], which states that most Web pages link to other pages that are similar in content.

2.1 Classifier-Guided Topical Crawlers

In situations where examples of relevant Web pages are available for topics, it is almost natural to involve classifiers when crawling. It is therefore surprising to observe that there has only been sporadic study on the role of classifiers in crawlers. Indeed besides the effort of Chakrabarti et al. [1999, 2002], few papers have investigated classifier-based crawling.

Chakrabarti et al. [1999] were the first to use a (Naive Bayesian) classifier to guide a topical crawler. The basic idea behind their crawler is to classify crawled pages with categories in a topic taxonomy. A user can mark some of the categories as "good" to indicate the information need. The crawler then uses the example URLs to build a Bayesian classifier that can find the probability $\Pr(c|p)$ that a crawled page p belongs to a category c in the taxonomy. We note that this focused crawler can be generalized and used without a taxonomy for crawling problems where only two categories (relevant/not relevant) are available [Chakrabarti et al. 2002]. Diligenti et al. [2000] suggested a variation of the above focused crawler and called it a context focused crawler. Unlike the focused crawler described above, this crawler uses a set of Naive Bayes classifiers that are trained to estimate the link distance between a crawled page and the relevant pages. More recently Johnson et al. [2003] suggested the use of an SVM classifier with a linear (1st degree) kernel for topical crawling. No comparisons were made in these studies to other classification algorithms or variations of the same algorithm (e.g., SVM with 2nd degree kernel).

Machine learning algorithms other than classifiers have been used to guide topical crawlers. For example, Chen et al. [1998] proposed a topical crawler that uses a genetic algorithm. A crawling algorithm based on reinforcement learning (RL) [Mitchell 1997] was suggested by Rennie and McCallum [1999]. Menczer and Belew [2000] detailed InfoSpiders, an evolutionary algorithm that generates a population of adaptive spiders that use Neural Networks with Q-learning.

We recognize classifier-guided topical crawlers as an interesting direction in the development of topical crawlers. This direction allows for flexible models of topics being discovered from example Web pages. As mentioned before, the main weakness of the current literature is the absence of a systematic study that compares various types of classifiers in the role of guiding a topical crawler. Also, studies that use large numbers of topics and make statistically robust comparisons while using such crawlers are lacking. Our goal is to fill some of these research gaps, while simultaneously suggesting new algorithms and seeking insights into the use of classifier-guided topical crawlers.

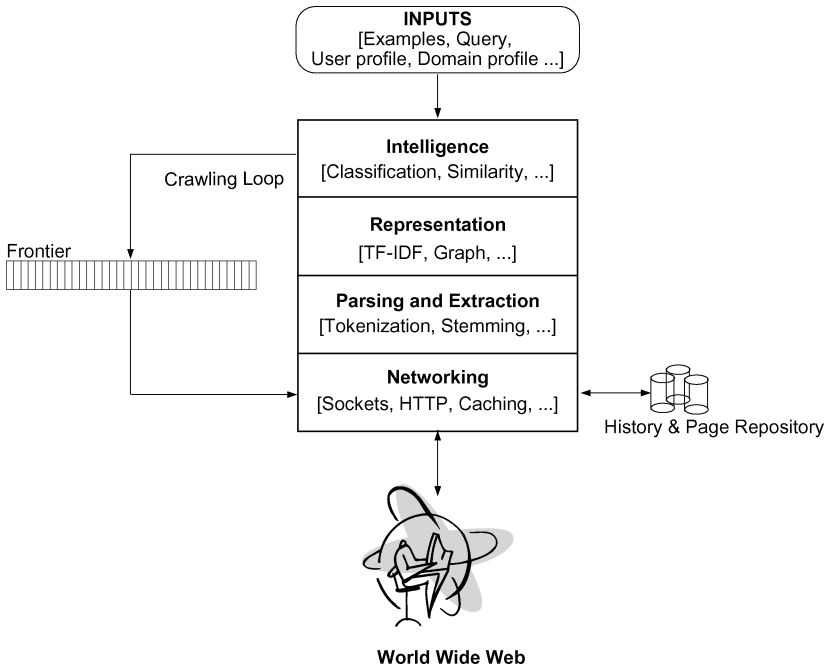

Fig. 1. High-level design of topical crawling infrastructure.

3. CRAWLER ARCHITECTURE

Figure 1 details our high-level layered design for a topical crawling infrastructure. Each layer is oblivious to the implementation details of other layers; this makes it easy to replace one implementation of a layer with another, as long as their interface is kept the same. In a *crawling loop* a URL is picked from the frontier (a list of unvisited URLs), the page corresponding to the URL is fetched from the Web, the fetched page is processed through all the infrastructure layers (see Figure 1), and the unvisited URLs from the page are added to the frontier. The *networking layer* is primarily responsible for fetching and storing a page and making the process transparent to the layers above it. The *parsing and extraction layer* parses the page and extracts the needed data such as hyperlink URLs and their contexts (words, phrases etc.). The extracted data is then represented in a formal notation (say a vector) by the *representation layer* before being passed onto the *intelligence layer* that associates priorities or scores with unvisited URLs in the page. We implement the high level layered design as multithreaded objects in Java. Each thread implements a sequential crawling loop that includes the layers shown in Figure 1. All of the threads share a single synchronized frontier.

When a classifier is used to guide a topical crawler, each thread in our implementation follows the steps shown in Figure 2. We use the Weka machine learning software for the implementation of the classifiers.[2] We chose Weka

[2]http://www.cs.waikato.ac.nz/ml/weka/

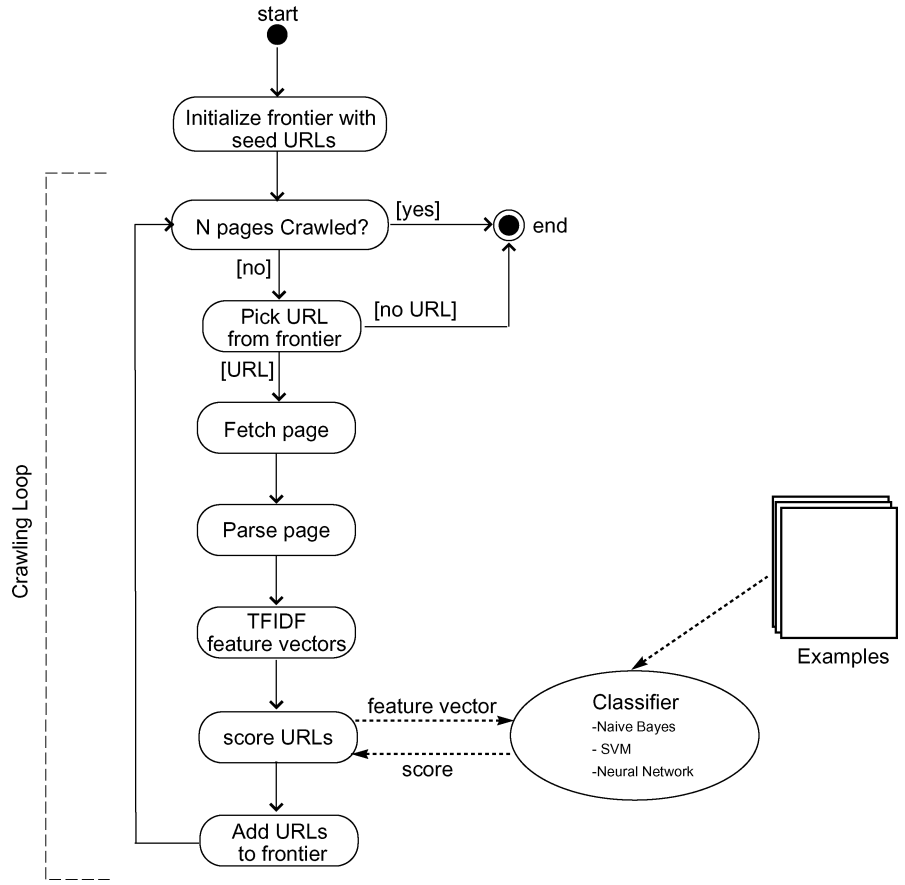

Fig. 2. Crawler model: details of a crawler thread and its interaction with the guiding classifier.

for our study because it is an open-source library of Java code that provides simple and consistent API for various classification algorithms. We could have used SVM^{light},[3] an implementation of SVM in C, along with locally developed software for Naive Bayes and Neural Networks. However, Weka offers the advantage of a consistent API. We appropriately interface our implementation with Weka so that classifiers can be used and replaced with ease and flexibility. In terms of the high-level design (see Figure 1) the change in classifiers amounts to changing only the intelligence layer while all of the other layers remain constant.

A topical crawler needs to process and represent Web content, apply intelligence (e.g., through a classifier), and maintain a frontier of URLs based on their priorities. All of these tasks require some amount of memory-resident data structures that are not necessary for a generic (non-topical) crawler. This additional memory consumption by a topical crawler may affect its scalability when compared with a generic crawler.

[3]http://svmlight.joachims.org/

4. CLASSIFICATION SCHEMES AND CLASSIFIERS

We differentiate between classification schemes and classifiers. A classification scheme is a general mechanism that can have many versions due to parameter and algorithmic variations. A classifier is a particular version of a scheme. Hence, we call SVM a classification scheme while we call a particular implementation of SVM, say with a linear kernel, a classifier. The purpose of a classifier is to assign an object (or its representation) to a class or a category. In the current experiments, the object is a Web page and the classes are "relevant" or "not relevant." A classifier makes the assignment based on historical or *training data* on the topic. This training data is a set of examples each of which includes an object, its representation, and its true class. A classifier learns from the examples through a training process. This form of learning is often called *supervised learning*. Once trained, the classifier can be used for assignment of previously unseen objects to classes. More formally, a two class (relevant/not relevant or positive/negative or $+1/-1$) classification problem where objects are represented in N dimensional real space ($\mathcal{R}^N$) is that of finding a function (classifier) $f : \mathcal{R}^N \rightarrow \{\pm 1\}$. The training data can be written as,

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m) \in \mathcal{R}^N \times \pm 1$$

where $\mathbf{x}_i$ is the vector representation of object i, and y_i is its class label [Müller et al. 2001]. Once the function is derived (classifier is trained) it draws a decision surface in the space of objects in an attempt to differentiate between the two classes.

Given a real world data set, such as pages and their true classes, no decision surface will be perfect in classifying unseen data. However, we would like the classifier to provide us with $\Pr(class = +1|\mathbf{x}_i)$, the probability that the page belongs to the relevant (positive) class given its representation. We can then associate this probability with the unvisited URLs within that page. This probability will serve as a score through which we can prioritize the frontier of URLs.

4.1 Representation, Training, and Usage

Our experiments are run over many topics that are obtained from the Open Directory Project (ODP).[4] The process used to extract topics from ODP is described later in Section 5. For each topic, we have positive and negative examples. The positive examples are Web pages that have been manually judged to be on the topic, and negative examples are randomly selected Web pages from other topics. We keep the number of negatives to be twice the number of positives. The positive and negative examples are represented in TF-IDF (term frequency-inverse document frequency) [Salton and McGill 1983] vector space. Hence, all of the examples are parsed and tokenized to identify the words within them. The stop-words are removed and the remaining words are stemmed using the Porter stemming algorithm [Porter 1980]. These stemmed words or *terms* from all of the negative and positive examples form our vocabulary $\mathcal{V}$ for the

[4]http://dmoz.org

topic. This vocabulary may differ across topics. Next, we represent the positive and the negative examples as feature vectors where each element in the vectors corresponds to a term in $\mathcal{V}$. Thus, we represent a Web page p as a vector $\mathbf{x}_p = (w_{1p}, w_{2p}, ..., w_{Np})$, where w_{sp} is the weight of the term s in page p and N is the size of $\mathcal{V}$. The term weights are TF-IDF values that are computed as:

$$w_{sp} = \underbrace{\left(0.5 + \frac{0.5 \cdot tf_{sp}}{\max_{s' \in T_p} tf_{s'p}}\right)}_{TF} \cdot \underbrace{\ln\left(\frac{|E|}{df_s}\right)}_{IDF} \tag{1}$$

where tf_{sp} is the frequency of the term s in page p, T_p is the set of all terms in page p, E is the collection containing the positive and the negative examples, and df_s (document frequency or DF) is the number of pages in E that contain the term s. Equation 1 corresponds to one of the standard weighting schemes in the SMART system [Salton 1971].

Both the positive and the negative example pages represented as TF-IDF based feature vectors are used for training a classifier. Once the classifier has been trained, it is used to estimate the $\Pr(class = +1|\mathbf{x}_i)$ where i is the crawled page and $\mathbf{x}_i$ is its vector representation. The space in which this page is represented remains confined to the vocabulary $\mathcal{V}$ for the topic. Also, the page i is represented in the same way as the positive and the negative examples, with term weights being computed as given in Equation 1. We will continue to use the df_s computed based on the collection E. Also, we note that no feature selection [Yang and Pedersen 1997] is performed beyond stoplisting and stemming. Feature selection would have added another dimension to our experiments, leading to potentially different crawl performances based on the feature selection technique used. We could have also explored various combinations of feature selection techniques and classifiers that perform well. However, we have left this dimension for future research.

We call the trained classifier the *crawling classifier* since we will use it to guide a topical crawler. We next describe the classification schemes that will be used in our experiments. We have picked three classification schemes that have been commonly and successfully used in various data mining and information retrieval applications. The use of different classifiers would lead to different costs in terms of training and classification times. The classification cost is paid at crawl time and hence must be judged relative to the time to download a Web page. We may expect the time to download a Web page to be much greater than the classification time.

4.2 Naive Bayes

A Bayesian classifier [Chow 1957; McLachlan 1992; Elkan 1997] is based on the well known Bayes formula. It assumes that given a class, the data ($\mathbf{x}$) is produced from a probability distribution (e.g., $p(\mathbf{x}|class = +1)$ if the class is positive). Hence, applying the Bayes formula, the posterior probability,

$\Pr(class = +1|\mathbf{x})$, can be computed as:

$$\Pr(class = +1|\mathbf{x}) = \frac{p(\mathbf{x}|class = +1) \cdot \Pr(class = +1)}{p(\mathbf{x})} \tag{2}$$

where $\Pr(class = +1)$ is the prior probability of the relevant class, and for a two class problem,

$$p(\mathbf{x}) = p(\mathbf{x}|class = +1) + p(\mathbf{x}|class = -1) \tag{3}$$

(we use $p(\cdot)$ for a pdf where the area under the function adds up to 1 and $\Pr(\cdot)$ for denoting probabilities). An added assumption in the Naive Bayes classifier is that the various components (attributes) of the data $\mathbf{x}$ are independently distributed. Hence, Equation 2 can be written as:

$$\Pr(class = +1|\mathbf{x}) = \frac{\prod_{j=1}^{N} p(x_j|class = +1) \cdot \Pr(class = +1)}{p(\mathbf{x})} \tag{4}$$

where x_j is the jth attribute of vector $\mathbf{x}$.

We compute the prior probabilities $\Pr(class = +1)$ and $\Pr(class = -1)$ based on how frequently a class appears in the training data. Since, $p(\mathbf{x})$ is just a normalization factor, the Naive Bayes' training process [McCallum and Nigam 1998; Lewis 1998] is all about estimating the underlying distributions of attributes given the class: estimating $p(x_j|class = +1)$ and $p(x_j|class = -1)$ for all j. The manner in which these distributions are estimated differentiates the two classifiers of this scheme available with Weka, that we have used.

Normal Density: In this case, we assume that the attribute values are normally distributed given the class. Hence, $p(x_j|class = +1)$ and $p(x_j|class = -1)$ for each x_j are normal distributions with some mean and variance. Then the learning problem is that of estimating the parameters—mean and variance. In other words, the learning problem is a *parametric* one. Since we have training examples from both the positive and the negative class, we can use those to estimate the needed parameters. We use the training data to derive the maximum likelihood estimates (MLE) of the mean and the variance of the distributions. The estimates will be the sample mean and variance of attribute values for each class where the sample is our training examples.

Kernel Density: A more powerful model is obtained when we remove the normality assumption and rely on nonparametric statistical methods for estimating the distribution for $p(x_j|class = +1)$ and $p(x_j|class = -1)$ [John and Langley 1995]. We use a nonparametric method where the $p(x_j|class = +1)$ is computed as:

$$p(x_j|class = +1) = \frac{1}{n_+} \cdot \sum_{i=1}^{n_+} k(x_j; \mu_+^i, \sigma_+^i) \tag{5}$$

where n_+ is the number of training examples that belong to the relevant class, x_{ji} is the j attribute of vector $\mathbf{x}_i$ (representation of the ith positive example), and k is a normal density function with mean $\mu_+^i = x_{ji}$ and standard deviation $\sigma_+^i = \frac{1}{\sqrt{n_+}}$ [John and Langley 1995]. A simple explanation for Equation 5 is that it takes an attribute value x_j from a vector representation of an object

(say a Web page) $\mathbf{x}$, and measures its average distance from the corresponding attribute value in each of the positive training examples and uses that as the $p(x_j|class = +1)$. $p(x_j|class = -1)$ is measured in a similar manner. The distance is computed using a gaussian (normal density) kernel (distance metric).

Note that once $\Pr(class = +1|\mathbf{x})$ and $\Pr(class = -1|\mathbf{x})$ have been computed, the assignment of class is simple. If $\Pr(class = +1|\mathbf{x}) > \Pr(class = -1|\mathbf{x})$, the class of $\mathbf{x}$ is relevant or positive, else it is not relevant or negative. We do not know of any previous work in topical crawling that uses Naive Bayes with kernel density estimation.

4.3 Support Vector Machines

Support vector machines (SVMs) have attracted much attention in the recent literature due to their good real-world performance and their theoretical underpinnings. Vapnik [1995] provides an authoritative treatment of SVMs. For a two class problem, the basic idea behind SVM learning is to find an optimal linear decision boundary (hyperplane) that minimizes the *risk* of erroneous classifications. This risk is defined in statistical terms and the training process amounts to solving a quadratic programming problem [Burges 1998; Platt 1999; Schölkopf et al. 1999; Cristianini and Schölkopf 2002; Schölkopf and Smola 2003]. Applying the Kurush-Kuhn-Tucker conditions [Wright and Nocedal 1999], it is found that the training examples that influence the optimal decision boundary are the ones that are closest to it. These training points are called the *support vectors* and the resulting optimal hyperplane classifier a *support vector machine*. The optimal hyperplane thus found can be written as:

$$g(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + w_0 = 0 \tag{6}$$

where $\mathbf{x}$ is a point on the plane, $\mathbf{w}$ are weights learned in the training process that determine the direction of the hyperplane, and w_0 fixes the position of the hyperplane in space [Theodoridis and Koutroumbas 2003].

A trick to extend the SVM learning to nonlinear decision boundaries is to convert the *input space* ($\mathcal{R}_N$) into a higher dimensional space such that a linear decision boundary in the new space corresponds to a nonlinear counterpart in the input space (see Figure 3) [Müller et al. 2001]. Hence, nonlinear separating surfaces can be created while solving a linear problem in the transformed space. To simplify, the desired transformation effect is achieved by just redefining *kernels* (distance metrics) without having to map each point from the input space to the transformed space. Note that the input space corresponds to the space where the objects are originally represented. The SVM implementation in Weka converts the output of an SVM ($g(\mathbf{x})$) for an object's vector representation $\mathbf{x}$ (that could lie on either side of the optimal hyperplane) into a score between 0 and 1 using a sigmoid function (see Equation 7). In addition it uses John C. Platt's sequential minimal optimization (SMO) [Platt 1999] for training an SVM. We do not change Weka's default values for variables such as complexity constant ($C = 1$) and accuracy tolerance (10^{-3}). We will use three versions of SVMs in our experiments corresponding to 1st, 2nd, and 3rd degree polynomial

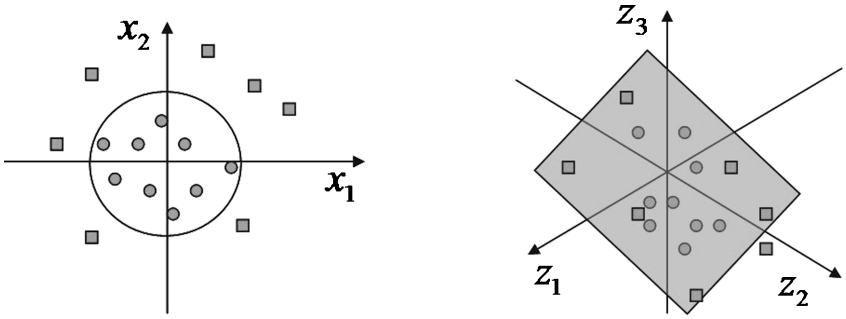

Fig. 3. Transformation (through ϕ_2) of a two dimensional input space to three dimensional transformed space. A circle in the input space corresponds to a hyperplane in the transformed space [Schölkopf et al. 1999; Müller et al. 2001].

kernels. To the best of our knowledge, no work in topical crawling has ever used SVM with 2nd and 3rd degree polynomial kernels.

4.4 Neural Networks

Neural Networks [Lippmann 1988; Rumelhart et al. 1994; Jain et al. 1996] are computational models "inspired" by their biological counterparts [McCulloch and Pitts 1943]. In our experiments we use a three layer feed-forward Neural Network [Hornik et al. 1989; Widrow 1990] as shown in Figure 4. The *input layer* nodes (circles) can take in a vector representation ($\mathbf{x}$) of an object and the *output layer* nodes represent the two classes (relevant or not relevant) in which $\mathbf{x}$ may fall. Each node in the input layer corresponds to a term in our vocabulary. The directed edges in the network (see Figure 4) that connect a pair of nodes have a weight associated with them. The node at the source of an edge provides an input that is then multiplied by the corresponding edge weight before reaching the destination node of that edge. At the destination node the weighted inputs received from all connected source nodes are summed up and passed through a sigmoid function to derive an output (which may in turn be an input to the next layer in the network). The sigmoid function is of the form:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{7}$$

where x is the sum of the weighted inputs from the source nodes in the previous layer. The threshold value typically seen in a sigmoid function is modeled as an additional weight connected to a source node with a constant output of 1. Weka uses the back-propagation [Rumelhart et al. 1986; LeCun 1986] learning strategy that uses the training data to adjust weights within the network in order to move it in the direction of steepest squared error reduction. Hence, this learning strategy is a form of *gradient descent* [Wright and Nocedal 1999]. The general weight change rule in back-propagation learning is:

$$\mathbf{w}_i^j \leftarrow \mathbf{w}_i^j + c \cdot \delta_i^j \cdot \mathbf{x}^{j-1} \tag{8}$$

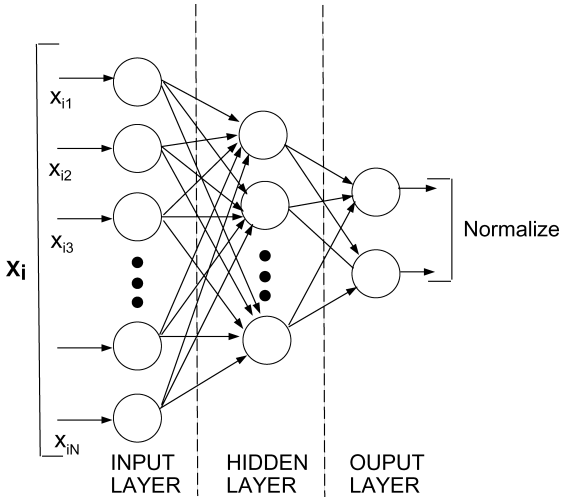

Fig. 4. A three layer feed-forward neural network: Input layer receives $\mathbf{x}_i$ corresponding to vector representation of a Web page. Output layer gives scores that are normalized to get the Pr($class = +1|\mathbf{x}_i$) and Pr($class = -1|\mathbf{x}_i$).

where $\mathbf{w}_i^j$ is a vector of weights connecting a node in the jth layer with the source nodes in the $j-1$th layer, c is the learning rate, δ_i^j controls the magnitude and sign of the adjustment to weights, and $\mathbf{x}^{j-1}$ is the input vector from the previous layer. All of these values are as they exist in the ith iteration of the training process. By default, the strategy we use here, Weka performs 500 training iterations or *epochs*.

The training process can be viewed as a nonlinear regression problem where the process estimates the constants and coefficients within the model (the Neural Network) [Duda et al. 2000]. A three layer feed-forward neural network with back-propagation learning is known to provide the universal approximation property [Hornik et al. 1989]; such a neural network can approximate any bounded continuous function.

We experiment with classifiers based on multiple versions of neural networks that differ in learning rates and number of hidden nodes. While the learning rate effects the magnitude of the adjustments in the network weights and thresholds at each step in the training process, the number of hidden nodes determines the complexity of the decision surface created. The literature does not provide any previous work that uses crawlers based on pre-trained multilayered neural networks.

Once a classifier has been trained it can be used in the intelligence layer of a topical crawler to provide scores for unvisited URLs based on their respective parent pages.

5. ODP TOPICS

We obtained topics for our crawls from the Open Directory Project (ODP). The following steps were used:

(1) Download the RDF format content file from the ODP Web site. The content file contains a listing of ODP topics (categories) and the external URLs or *ODP relevant set* corresponding to each topic. The ODP relevant set consists of URLs that have been judged relevant to the topic by human editors of ODP.
(2) Filter the content file to remove topics such as `Regional`, `International`, and `Adults`. This is needed to avoid many semantically similar topics, sites involved in search engine spamming, as well as pages with non-English content. Also filter to remove the topics that have less than 40 URLs so that we have topics with a critical mass of URLs for training and evaluation.
(3) From the remaining topics (more than 7000) randomly select a desired number of topics and the associated ODP relevant sets. Due to the large number of available topics, we do not expect random sampling to provide us with a cluster of similar topics. It is possible that good quality pages, such as the ODP pages, are better-connected and hence more reachable by crawlers than random pages on the web. However, this aspect will not bias this research in favor of one topic over another, or in favor of one crawler over another.
(4) Divide the ODP relevant set for a selected topic into two random disjoint subsets. The first set is the seeds. This set of URLs will be used to initialize the crawl as well as provide the positive examples to train the classifiers. The second set is the *targets*. It is a holdout set that will not be used either to train or to seed the crawler. The seed set contains 20 URLs. The rest of the (20 or more) external URLs make up the targets. These targets will be used only for evaluation of crawl performance.

Table I shows sample topics and their corresponding seeds and targets. As explained before, for training a two category classifier we need both positive and negative examples. The positive examples are pages corresponding to the seed URLs for a given topic. The negative examples are a random subset of positive examples for the other topics in the set of selected topics; it is twice in number as compared to the positive examples. Obtaining appropriate negative examples as well as the appropriate proportion of positive and negative examples is a hard problem on the Web. We have opted for a simple technique while maintaining the number of negative examples to be larger than the positive examples.

6. PERFORMANCE METRICS

The output of a crawler l for a given topic can be written as a temporal sequence $\mathcal{S}_l = \{u_1, u_2, \ldots, u_M\}$ where u_i is the URL for the ith page crawled and M is the maximum number of pages crawled. In order to measure the performance of the crawler l we need a function f that maps the sequence $\mathcal{S}_l$ to a sequence $\mathcal{R}_l = \{r_1, r_2, \ldots, r_M\}$ where r_i is a scalar that represents the relevance of the ith page crawled. Once we have a sequence $\mathcal{R}_l$ we may summarize the results at various points during the crawl. However, evaluation of topical crawlers is a challenging problem due to the unavailability of the complete relevance

Table I. Sample Topics—ODP Topic Name, Seeds, and Targets

topic name	seeds	targets
Science/Chemistry/Education	http://haberchemistry.tripod.com/ http://www.creative-chemistry.org.uk/ http://www.heptune.com/passchem.html http://chemistry.org/portal/Chemistry/ ...	http://www.syvum.com/squizzes/chem/ http://seaborg.nmu.edu/ http://www.towson.edu/saacs/ http://www.secondlaw.com/ http://www.spusd.k12.ca.us/chemmybear/ ...
Business/Food_and_Related_Products/Dairy/Ice_Cream	http://www.paolosgelato.com/ http://www.perrysicecream.com/ http://www.haganicecream.com/ http://www.spaghettiicecream.com/ ...	http://www.carvel.com/ http://www.allscream.com/ http://www.greensicecream.com/ http://www.williesicecream.com/ http://www.jackjillicecream.com/ ...
Computers/Security/Anti_Virus/Products	http://www.sybari.com/ http://www.kaspersky.com/ http://www.virusarrest.com/ http://www.norman.com/ ...	http://www.grisoft.com/ http://www.mwti.net/ http://www.pspl.com/ http://www.ravantivirus.com/ http://www.centralcommand.com/ ...
Society/Religion_and_Spirituality/Divination/Astrology/Vedic	http://www.astromandir.com/ http://www.eastrovedica.com/ http://www.macrovedic.com/ http://www.mywebastrologer.com/ ...	http://www.jyotish-yagya.com/ http://www.sgandhi.co.uk/ http://enmag.org/jyotish.htm http://www.indianastrology.com/ http://www.shyamasundaradasa.com/ ...

set for any topic or query. Also, manual relevance judgement for each of the crawled pages (manually created function f) is extremely costly in terms of man-hours when we have millions of crawled pages spread over more than one hundred topics. Even if we sample for manual evaluation, we will have to do so at various points during a crawl to capture the temporal nature of crawl performance. Hence, even such samples would be costly to evaluate over a hundred topics. Hence, the two standard information retrieval (IR) measures *recall* and *precision* can only be estimated using surrogate metrics. Below we describe *harvest rate*, which we use as an estimate of precision and *target recall*, which we use as an estimate of recall.

Harvest Rate: It estimates the fraction of crawled pages that are relevant. How do we judge the relevance of a crawled page? We depend on multiple classifiers to make this relevance judgement. These *evaluation classifiers* are essentially the same as those used for crawling but they are trained on a larger set (at least double in number) of examples. For each topic, we take one classifier at a time and train it using the pages corresponding to the entire ODP relevant set (instead of just the seeds) as the positive examples. The negative examples are obtained from the ODP relevant sets of the other topics. The negative examples are again twice as many as the positive examples. These trained classifiers are called evaluation classifiers to distinguish them from crawling classifiers that are used to guide the crawlers. Once we have a group of evaluation classifiers, we judge each of the pages crawled for the topic and then consider it to be relevant if at least half of the classifiers judge it to be relevant. Thus a majority vote amongst the evaluation classifiers (this will be further illustrated in Section 7) judges the relevance of a page and gives us the needed function f. The harvest

rate, $\mathcal{H}(t)$, after crawling the first t pages is then computed as:

$$\mathcal{H}(t) = \frac{1}{t} \sum_{i=1}^{t} r_i \tag{9}$$

where r_i is the binary (0/1) relevance score for page i based on a majority vote amongst the evaluation classifiers. The harvest rate is computed at different points (t) during the crawl. We then average the harvest rate at these points over all the topics in the test bed. Along with the average, we also compute the ± 1 standard error. With the computed information we can plot the trajectory of average harvest rate for a crawler over time (approximated by t crawled pages). Our harvest rate metric while building on a similar metric suggested by Chakrabarti et al. [1999], differs from it in two aspects. First, we use several instead of a single classifier to judge the relevance of the crawled pages. Second, we use classifiers that are trained on a larger set of examples than the classifier used for crawling. In contrast, Chakrabarti et al. [1999] used exactly the same classifier (trained on the same examples) to judge the crawled pages and also to guide the crawler. We note that our harvest rate metric is analogous to similarity-based precision measures previously suggested by us [Srinivasan et al. 2005]. However, the similarity is measured here using a set of classifiers. Such a similarity measurement is appropriate in the context of classifier-guided topical crawlers.

Target Recall: Target recall [Pant and Menczer 2003; Srinivasan et al. 2005] estimates the fraction of relevant pages on the Web that are downloaded by the crawler. In the absence of a known relevant set, we treat the recall of the target set, target recall, as an estimate of actual recall. If targets are a random sample of the relevant pages on the Web, then we can expect target recall to give us a good estimate of the actual recall. While we cannot expect targets to be a true random sample of the relevant pages, they do provide a reasonable basis for comparative analysis of different crawlers. As with harvest rate, we also average the target recall over all the topics in the test bed. This give us the average target recall trajectory over time.

7. EXPERIMENTS

7.1 Selecting the Best Classifier within each Scheme

We begin with a preliminary experiment to identify the better version for each classification scheme. We test the following versions of the three classification schemes:

(1) Naive Bayes—normal density and kernel density;
(2) Support Vector Machines (SVMs)—1st, 2nd, and 3rd degree kernels;
(3) Neural Networks—2,4, and 16 hidden nodes and learning rates of 0.1, 0.2, and 0.4. We first find the better setting for the number of hidden nodes while keeping the learning rate at 0.2. Then we fix the number of hidden nodes to the better value and determine the best learning rate among the

three values. In this way we first find the better architecture (number of hidden nodes) and then choose the better learning rate.

Hence, we test 11 crawlers guided by 11 different classifiers. In this preliminary experiment we crawl 10, 000 pages for 10 topics that are obtained as described in Section 5. For measuring the harvest rate we use evaluation classifiers built using the 2 versions of Naive Bayes, the 3 versions of SVMs, and only 3 out of the 6 versions of Neural Networks. The 3 that we use correspond to the different numbers of hidden nodes (2,4,16) with a fixed learning rate (0.2).

7.2 Comparing Classification Schemes

Once the better classifier for each scheme has been identified in the previous experiment, we conduct a more rigorous test of crawlers built on these select classifiers. Hence, this experiment will have three crawlers, one for each scheme. Since Naive Bayes classifiers have been previously used in the crawling literature [Chakrabarti et al. 1999; Diligenti et al. 2000; Pant et al. 2004b], our best Naive Bayes crawler may be considered a reasonable baseline for comparison. We perform this experiment over the 100 topics (distinct from the initial set of 10 topics), and 10, 000 pages are crawled for each topic.

7.3 Comparing Classification Schemes with Few seeds

A related question that we address, concerns the number of seeds used by a crawler. What if there are only a few seeds available? Will the order of performances between the crawlers change? To answer these questions we repeat the previous experiment with just 5 seeds for each topic. We note that fewer seeds also means fewer examples used to train the guiding classifier. For this experiment a random subset of seeds of size 5 is created from the initial seed set of 20. However, when evaluating the runs we keep the targets for a given topic the same as before.

7.4 Exploring the value of Adaptation

The experiments described thus far use classifiers that are trained before the crawl begins. Moreover, the crawler learns nothing new during the time of the crawl. As an extension we also explore the addition of new pseudo examples to the original examples after crawling for a short time. We do this for the case where we start with just 5 seeds. We also limit this experiment to the "better" crawler from our comparison of classification schemes. This crawler will crawl 1000 pages and then pick the top 15 crawled pages (based on scores provided by the crawling classifier) as pseudo positive examples. Additionally, the bottom 30 crawled pages will be treated as pseudo negative examples. These pseudo examples $(15 + 30)$ along with the original examples $(5 + 10)$ are used to retrain the classifier before crawling continues for the remaining 9000 pages. Note that with the pseudo-examples, the crawler has a total of 20 positive and 40 negative examples. These numbers are the same as those for the experiment where we begin with 20 seeds. In all but the preliminary experiments where we select the best classifiers for each of the three schemes, harvest rate will be based on

three evaluation classifiers corresponding to the best classifiers from each of the three classification schemes studied.

We note that many of the algorithms that we test have never been tried before. For example, to the best of our knowledge, no work in topical crawling has ever used SVM with 2nd and 3rd degree kernels or Naive Bayes with kernel density estimation. Results of crawlers that use pretrained (from positive and negative examples) three-layered neural networks are also missing from the literature. Although the Infospiders algorithm uses a neural network with Q-learning to train during the time of crawl, it does not use a pretrained classifier [Menczer and Belew 2000].

8. RESULTS

8.1 Preliminary Experiments

Figure 5 shows the results of preliminary experiments for multiple versions of each classification scheme. The goal is to select the best version for each scheme. The horizontal axis in all of the plots is time approximated by the number of pages crawled. The vertical axis shows a performance metric—average harvest rate or average target recall. The error bars are also presented for each data point.

Since the preliminary experiments are conducted over just 10 topics the results in general do not allow us to identify a significant winner among the choices for each classification scheme. However, we depend on the average results to identify the classifier within each scheme that is as good as or better than any other from the scheme. We find that for Naive Bayes (Figure 5 (a) and (b)), the classifier with kernel density performs the best on both of the performance metrics. However, only the average target recall plot (Figure 5 (b)) shows a significant result between the two versions (we performed a one tailed t-test [Hogg et al. 2004] at $\alpha = 0.10$). Similarly for the SVMs, the crawler with 1st degree kernel is better than others (although not significantly) on both of the metrics (Figure 5 (c) and (d)). Results for Neural Networks, however, require a closer examination. The Neural Network with 4 hidden nodes performs better than the other two versions on average harvest rate for most of the 10, 000 page crawl (Figure 5 (e)). On average target recall the Neural Network with 4 (NN4) hidden nodes is outperformed by the Neural Network with 16 (NN16) hidden nodes by the end of the crawl (Figure 5 (f)). To make our choice, we look at the p-values [Hogg et al. 2004] of two different one-tailed t-tests. $h_{c,p}$ is the average harvest rate for a crawler based on classifier c after crawling p pages. Similarly, $r_{c,p}$ is the average target recall for a crawler based on classifier c after crawling p pages. Also, H_0 is the null hypothesis and H_1 is the alternative hypothesis.

(1) Test PH - H_0: $h_{NN4,10000} \leq h_{NN16,10000}$, H_1: $h_{NN4,10000} > h_{NN16,10000}$.
(2) Test PR - H_0: $r_{NN4,10000} \geq r_{NN16,10000}$, H_1: $r_{NN4,10000} < r_{NN16,10000}$.

We find the p-value of test PH to be 0.29 while that for test PR to be 0.43. While neither of the two p-values show statistically significant results, the alternative hypothesis of test PR is less likely as compared to the alternate hypothesis for

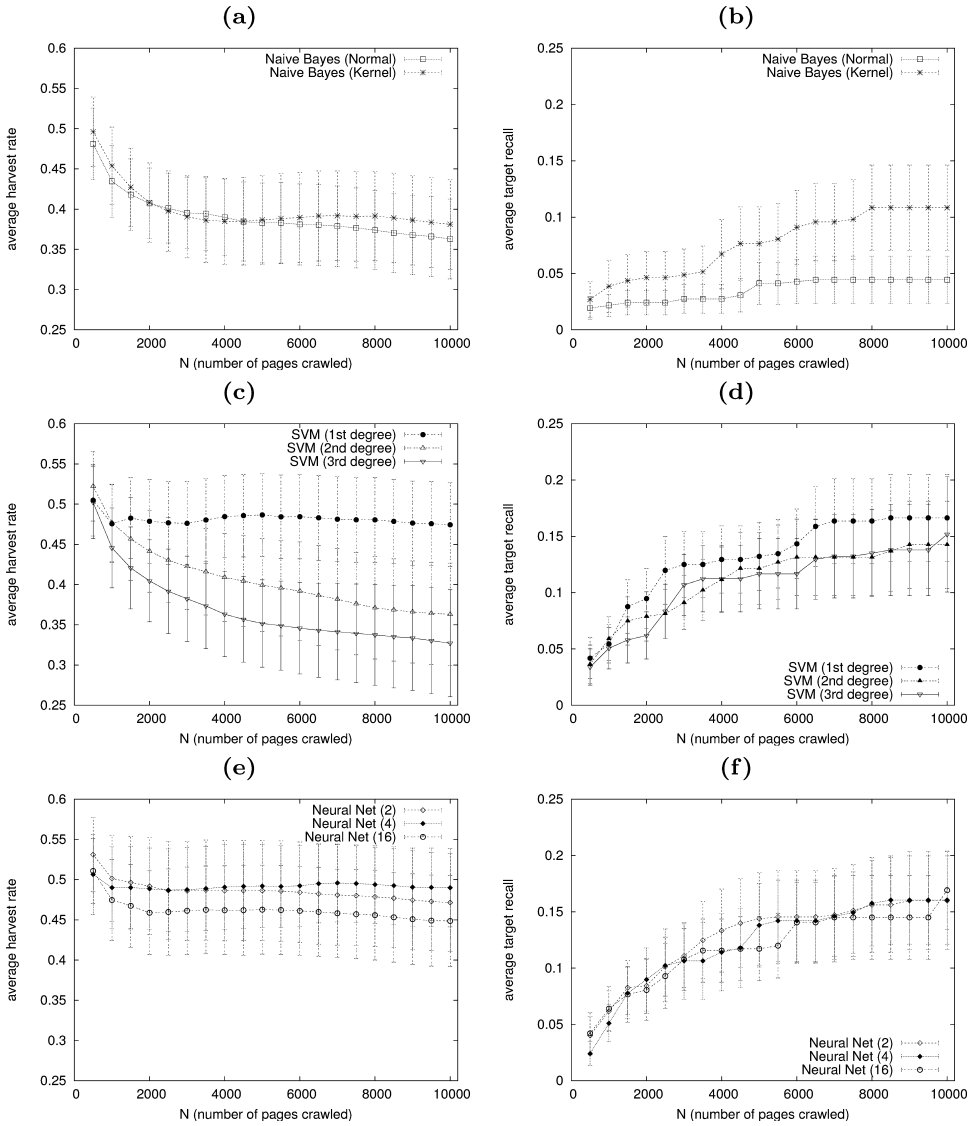

Fig. 5. Preliminary Experiments: Naive Bayes (a) average harvest rate (b) average target recall; SVM (c) average harvest rate (d) average target recall; Neural Network (e) average harvest rate (f) average target recall. The errorbars in these and following plots show ± 1 standard error.

test PH. We also note that by the end of the crawl, NN2 is worse than NN4 on both average harvest rate and average target recall. Hence, we choose NN4 as the best alternative among the three.

Having found the best number of hidden nodes (among those tested) while keeping the learning rate fixed at 0.2, we next vary the learning rate (0.1, 0.2, 0.4), while keeping the number of hidden nodes fixed at 4. Figures 6(a) and (b) show the performance of the crawlers based on Neural Networks obtained

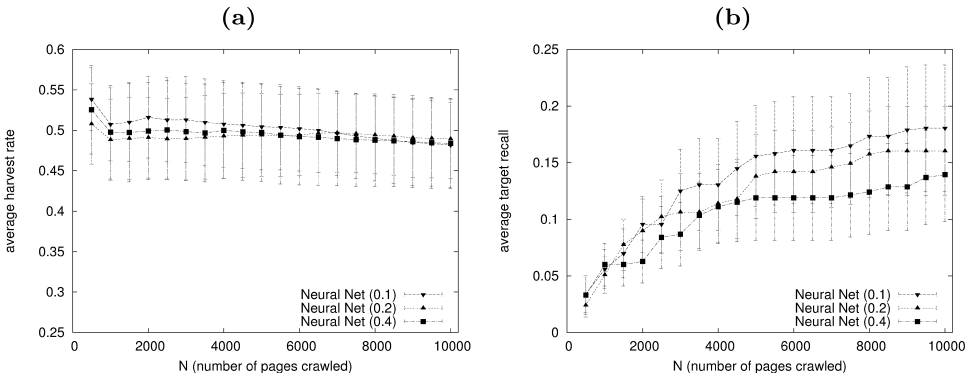

Fig. 6. Preliminary Experiments: Neural Network with different learning rates.

in this manner. Performing t-tests similar in nature to ones described earlier (this time between the Neural Network with learning rate of 0.1 vs. the Neural Network with learning rate of 0.2 and 0.4), we find that Neural Network with a learning rate of 0.1 seems to be the best choice. We note that we conducted the comparisons between Neural Networks with different numbers of hidden nodes based on a learning rate of 0.2 and now we find a learning rate of 0.1 to be better. However, we do not feel the need to repeat the experiments for the appropriate number of hidden nodes, since the differences between performances of crawlers based on Neural Networks with different learning rates is very small.

Based on these results we select the following versions as representative of the corresponding classification schemes for further experimentation:

(1) Naive Bayes with kernel density (henceforth called Naive Bayes);
(2) SVM with 1st degree kernel (henceforth called SVM);
(3) Neural Network with 4 hidden nodes and learning rate of 0.1 (henceforth called Neural Network).

We call the Naive Bayes guided crawler the NB crawler, the SVM-guided crawler the SVM crawler, and the Neural Network guided crawler the NN crawler.

8.2 Comparing Classification Schemes

These experiments are over the 100 topics with 10, 000 pages being crawled for each topic. Figures 7 (a) and (b) show the performance of the different crawlers. We find that the NB crawler for most of the crawl (up till the end) performs significantly poorly as compared to SVM crawler or NN crawler, both on average harvest rate as well as average target recall. We perform the following one tailed t-tests to measure the significance of performance differences:

(1) Test FH1 - H_0: $h_{NN,10000} \leq h_{NB,10000}$, H_1: $h_{NN,10000} > h_{NB,10000}$.
(2) Test FR1 - H_0: $r_{NN,10000} \leq r_{NB,10000}$, H_1: $r_{NN,10000} > r_{NB,10000}$.
(3) Test FH2 - H_0: $h_{SVM,10000} \leq h_{NB,10000}$, H_1: $h_{SVM,10000} > h_{NB,10000}$.
(4) Test FR2 - H_0: $r_{SVM,10000} \leq r_{NB,10000}$, H_1: $r_{SVM,10000} > r_{NB,10000}$.

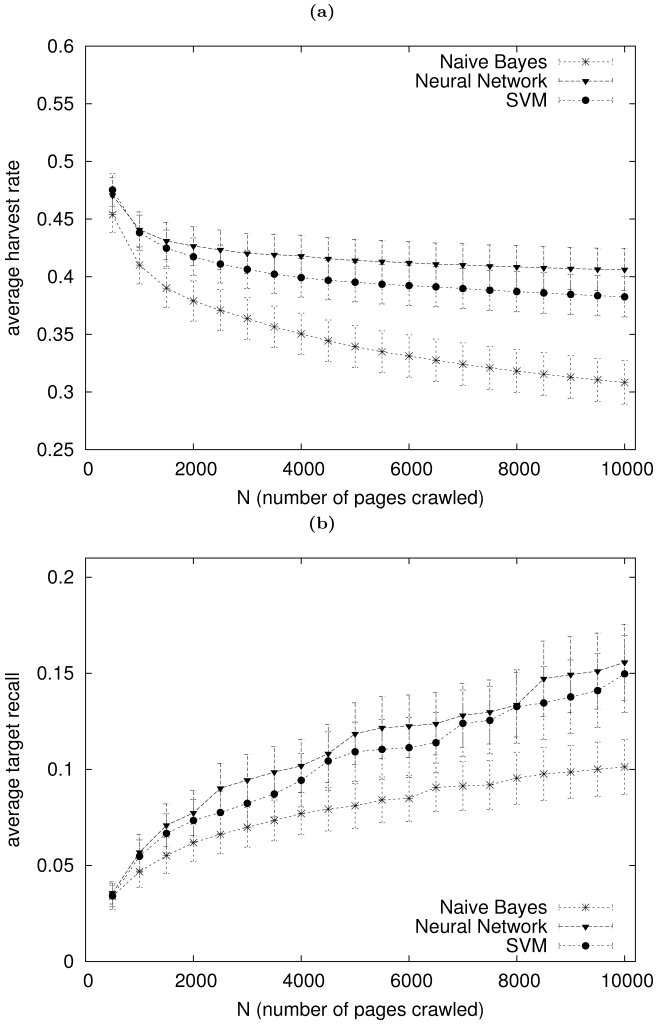

Fig. 7. Comparing Classification Schemes: (a) Average harvest rate (b) Average target recall.

(5) Test FH3 - H_0: $h_{NN,10000} \leq h_{SVM,10000}$, H_1: $h_{NN,10000} > h_{SVM,10000}$.
(6) Test FR3 - H_0: $r_{NN,10000} \leq r_{SVM,10000}$, H_1: $r_{NN,10000} > r_{SVM,10000}$.

Table II shows the p-values for these tests. We find that the first four tests reject the null hypothesis at $\alpha = 0.05$ significance level. Hence, both the SVM crawler and the NN crawler significantly outperform the NB crawler. The differences between SVM and NN are statistically insignificant.

We also compute the average training times for each of the classifier-guided crawlers. Table III shows the average training times with ± 1 standard errors. Based on the results on average harvest rate, average target recall, and the average training times, SVM seems to be the best choice of the three classifiers for guiding topical crawlers since SVM performs as well as Neural Network while requiring lower training time.

Table II. p-Values for Various Tests of Significance Between Performances of Classifier-Guided Topical Crawlers

Test	*p-value*	*Result* ($\alpha = 0.05$)
FH1	0.0001	H_0 rejected in favor of H_1
FR1	0.0134	H_0 rejected in favor of H_1
FH2	0.0022	H_0 rejected in favor of H_1
FR2	0.0254	H_0 rejected in favor of H_1
FH3	0.1755	H_0 is not rejected
FR3	0.4155	H_0 is not rejected

Table III. Average Training Time (in seconds) for Classifiers

Naive Bayes	*SVM*	*Neural Network*
0.310 ± 0.005	0.518 ± 0.009	56.637 ± 1.988

Table IV. Classification Time vs. Download Time (in seconds) for a Single Thread

Average Classification Time			Average Download Time
Naive Bayes	*SVM*	*Neural Network*	*for all crawlers*
0.0068	0.0002	0.0031	0.8767

We also note that the average time to classify a page (by a thread) during the crawl time is much smaller as compared to the average time to download a page (by a thread) for all of the crawlers (see Table IV). Hence, none of the classifiers appreciably change crawling time. In other words differences in classification time make little difference in the overall time consumed by a crawler. We also observe that it would be misleading to treat the classification time alone as the cost paid by a crawler since it is a very small fraction of the overall time consumed in downloading and processing a Web page.

8.3 Comparing Classification Schemes with Few Seeds

Figures 8 (a) and (b) show the performance of NB, SVM, and NN crawlers initialized with just 5 seeds. The purpose of the experiment was to observe the effects of a very small seed set on the performance of the crawlers. We note that the order of performance between the crawlers does not change. However, the significance of performance differences between the crawlers changes. In particular, SVM crawler is no longer significantly better than NB crawler on both of the metrics. The NN crawler, however, still significantly outperforms the NB crawler on average harvest rate. The performance difference on average target recall is not as significant (only significant at $\alpha = 0.1$). As we may expect, all of the topical crawlers perform significantly worse with 5 seeds, both on average harvest rate and average target recall, when compared with 20 seeds (cf. Figure 8 and Figure 7). It may be interesting to tease out the effects of a smaller set of seeds from the effect of a smaller set of examples. We have not explored this issue. In practice, it would be hard to imagine why someone would not provide all of the available positive examples as seeds.

8.4 Exploring the Value of Adaptation

The goal of this experiment is to explore a mechanism for finding pseudo examples after a short crawl in an attempt to provide more information to a crawler

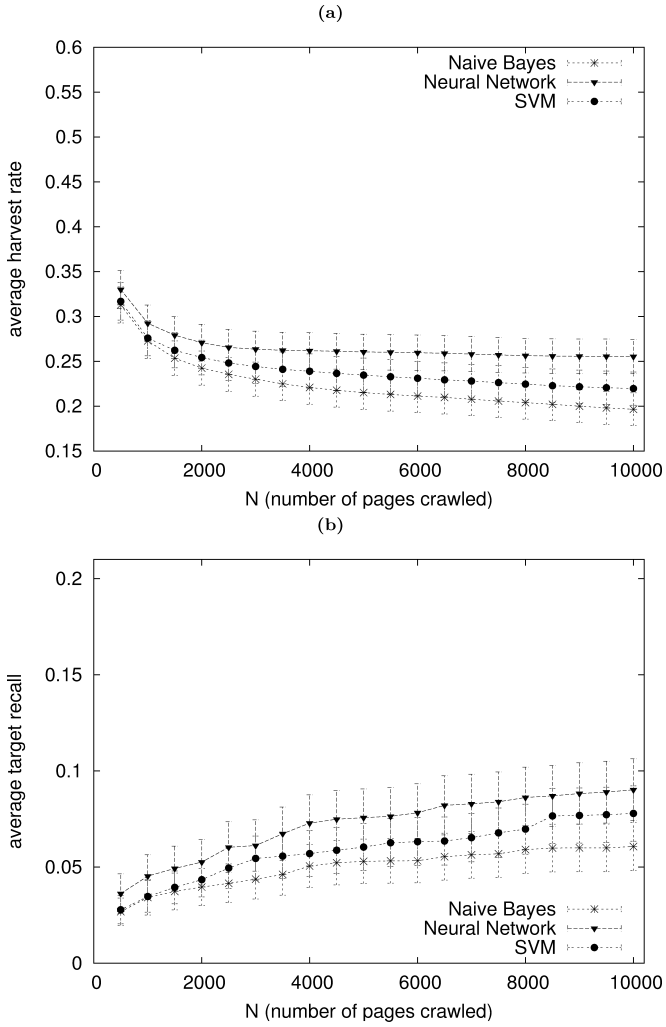

Fig. 8. Comparing Classification Schemes with Few (5) Seeds: (a) Average harvest rate (b) Average target recall.

that was initialized with very few examples. Figures 9 (a) and (b) show the effect of retraining a classifier after crawling 1000 pages. The crawler that retrains is called SVM Adaptive. We compare this crawler with the SVM crawler used in the previous experiment with few seeds (5 seeds). We find that adaption in this manner has little if any effect on performance. Although, on average target recall, the SVM Adaptive crawler seems to slightly outperform SVM during the latter part of the crawl (Figure 9 (b)), the difference is statistically insignificant. For average harvest rate (Figure 9 (a)), the SVM crawler on an average performs better than SVM Adaptive, but with insignificant difference. We may conclude from the plots that adding pseudo examples in this way does not provide the crawler with any new information that would help it improve its performance.

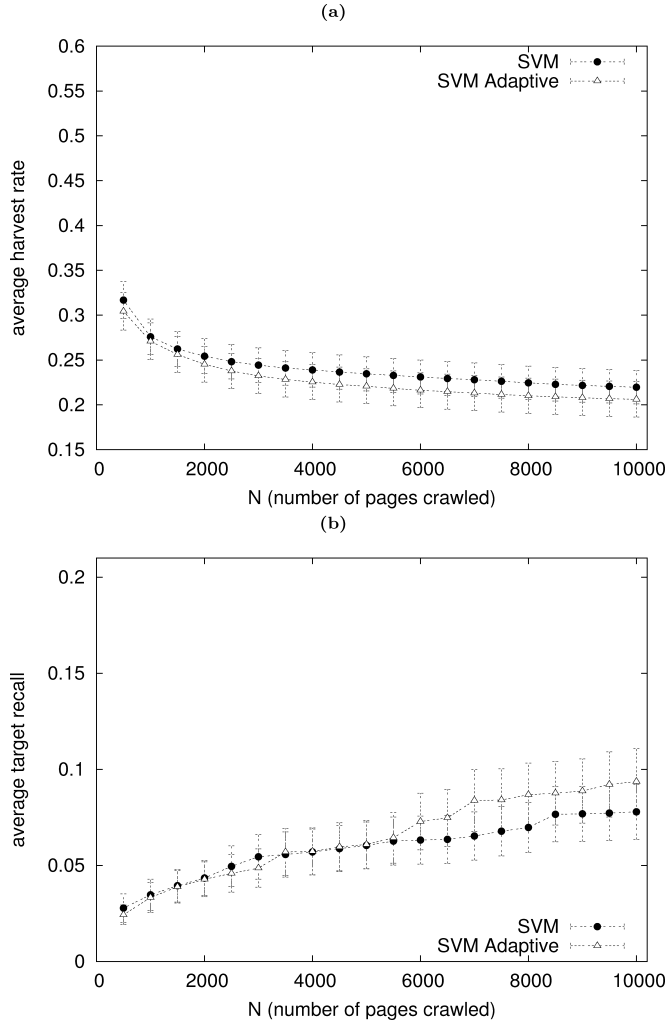

Fig. 9. Adaptation through pseudo examples: (a) Average harvest rate (b) Average target recall.

9. ANALYSIS

We further analyze our results in order to better understand the strengths and weaknesses of different topical crawlers and obtain insights into their behavior. The analysis presented now is conducted in a post hoc style using in all cases the 100 topic, 10, 000 pages per topic crawl data from the experiments.

9.1 Weakness of Naive Bayes

We observe the NB crawler performing poorly when compared with the SVM or the NN crawler. Could this be because Naive Bayes is a poor classifier when compared with the other two? To explore this question, we take the three classifiers and perform 5-fold cross-validation runs for each of the 100 topics. The cross-validation for a given topic is performed over its entire ODP relevant set

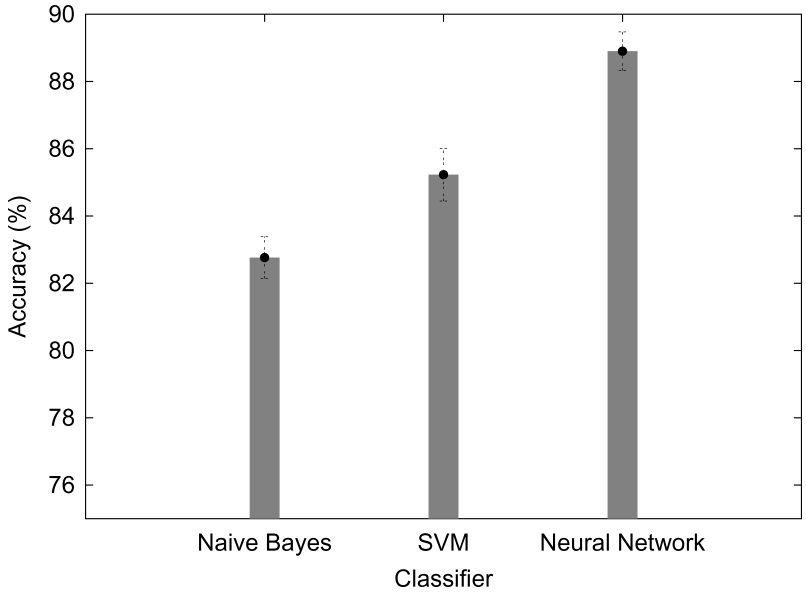

Fig. 10. Predictive accuracy of the three classifiers.

(as the positive instances/examples) that includes the seed set and the hold-out set used for computing the target recall. The negative examples for cross-validation are a random sample of ODP relevant sets from the other topics in our collection. For a given topic, a 5-fold cross-validation involves randomly splitting the set of positive and negative examples into 5 equally sized subsets. A classifier is trained using, in turn, 4 of the 5 subsets and tested on the fifth. In particular, we measure the accuracy of predictions of the trained classifier on the instances in the fifth set. Figure 10 shows the accuracy of classifiers based on these 5-fold cross-validation runs. Note that the accuracies are averages over the 100 topics.

We find that Figure 10 reflects the crawler performances in the experiment that compares the classification schemes (Figure 7 (a) and (b)). However, accuracy achieved by the SVM classifier (85.23 ± 0.78) is closer to the Naive Bayes classifier (82.76 ± 0.62) than to the Neural Network classifier (88.91 ± 0.58). In contrast, Figure 7 (a) and (b) shows that the SVM crawler significantly outperforms the NB crawler and also that the SVM and NN crawlers are not significantly different. Is there something fundamentally weak with Naive Bayes that leads to its poor performance in guiding a topical crawling? As explained before, the central issue in topical crawling is an effective prioritization over the frontier. What is the nature of priorities assigned to URLs by the different crawlers? To answer this question, we plot histograms that give us a sense of the distribution of scores given to crawled pages by the different crawling classifiers during the crawl. Figure 11 shows such plots for each of the three crawling classifiers used in the experiment that compares classification schemes. We find that the page scores assigned by Naive Bayes are extremely skewed, tending to be close to or equal to 0 or 1 (Figure 11 (a)). Hence, the NB crawler tends to lump

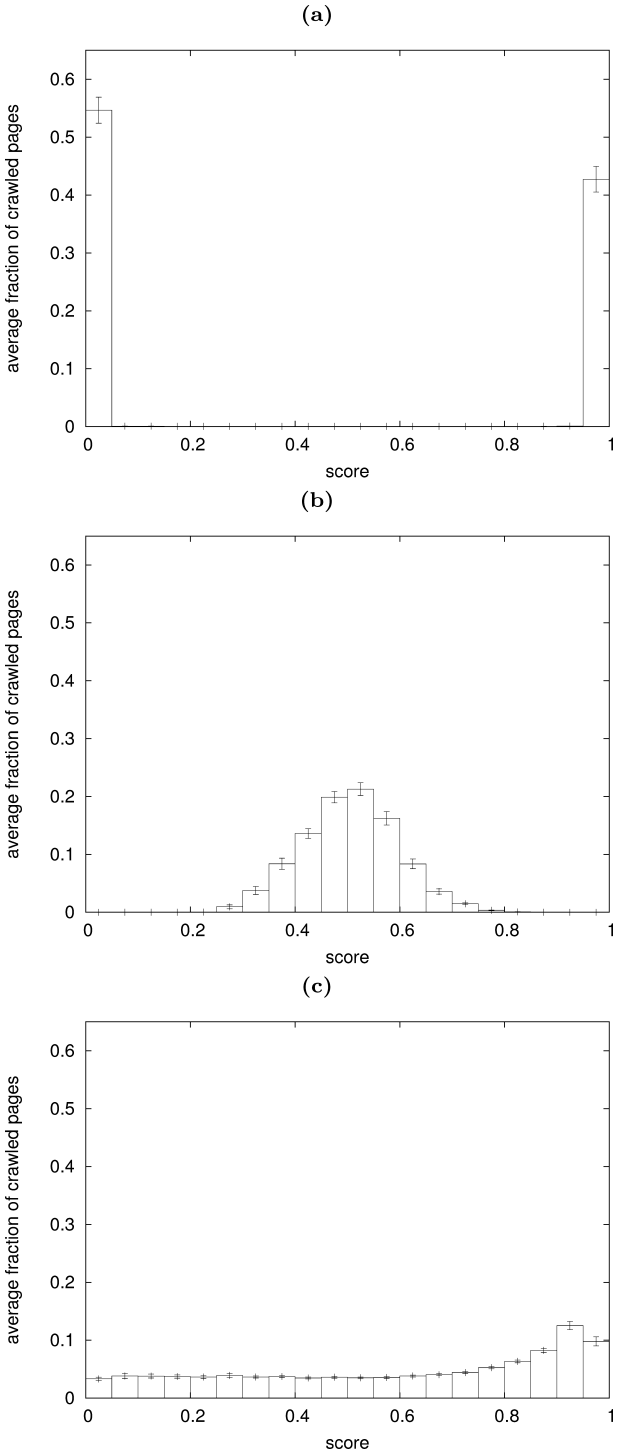

Fig. 11. Page score distributions: (a) Naive Bayes (b) SVM (c) Neural Network.

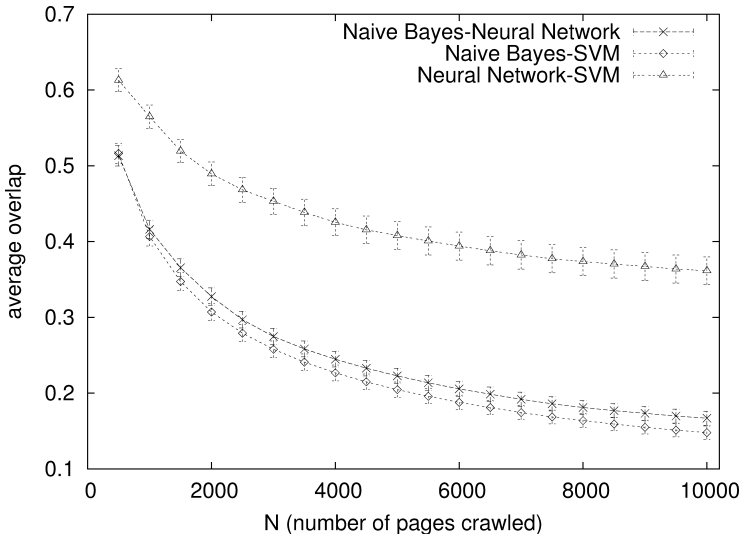

Fig. 12. Average pairwise URL overlap between SVM, NN, and NB crawler at various points during the 10, 000 page crawl. The overlap is measured relative to N (number of pages crawled).

crawled pages as "relevant" or "irrelevant", without providing enough shades of gray. This extreme skewness must affect its ranking and hence prioritization capabilities. Since page scores are assigned to the unvisited URLs within them, the skewness fails to effectively prioritize the frontier, hence explaining the particularly weak performance by the NB crawler. Naive Bayes in general (not just the version we use) is known to provide extremely skewed scores in high dimensional spaces [Chakrabarti et al. 2002]. Our classification problem is high dimensional since each term corresponds to a dimension. It is interesting that while Naive Bayes has been previously used for crawling [Chakrabarti et al. 1999; Diligenti et al. 2000], the literature does not investigate the problem due to skewed scores. Also as mentioned before, we do not know of any work in the literature that compares Naives Bayes with other classification schemes for topical crawling.

The histograms of Neural Network and SVM (Figure 11 (b) and (c)), although different from each other, behave well in the sense that they provide different shades of gray to the crawled pages.

9.2 Similar Performance Yet Dissimilar Pages

Turning our attention now to the SVM and NN crawlers we observe that they perform the same on both of the performance metrics (Figure 7 (a) and (b)). Does this mean that they cover the same subspaces on the Web? To answer this question we look at the overlap between the two crawlers based on common URLs crawled by both at different points during the crawl. This overlap (Figure 12) is averaged over all of the 100 topics. We note that the SVM and NN crawlers have a significantly higher percentage of common URLs crawled when compared with their overlap with the NB crawler. By the end of the 10, 000 page

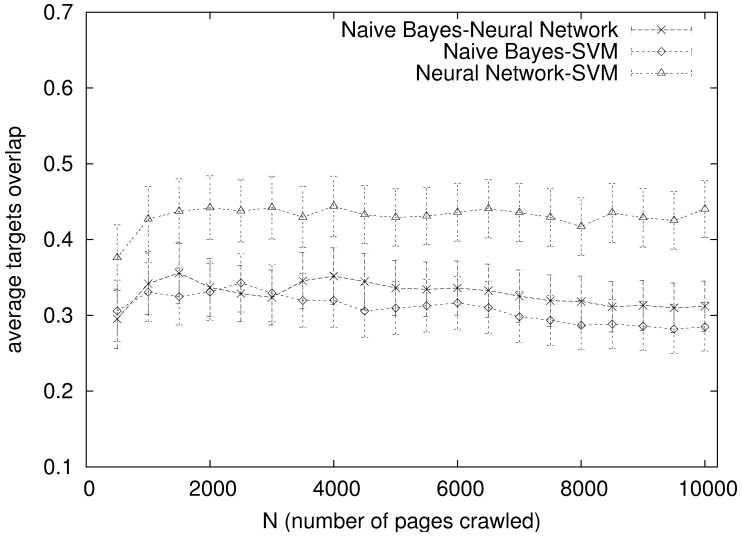

Fig. 13. Average pairwise overlap of targets between SVM, NN, and NB crawlers at various points during the 10, 000 page crawl. The overlap is measured relative to total targets obtained by the pair of crawlers after crawling N pages.

crawl, however, less than 40% of the pages are in common between the SVM crawler and the NN crawler. Hence, while they perform similarly on the metrics, they are exploring different subspaces of the Web. This result has interesting implications for an application designer—the choice of crawler will have an effect on the end application even if the collections created seem equally good.

Perhaps more importantly, Figure 13 shows a plot that measures the overlap between the crawlers based on the common targets retrieved from the Web. Again, the SVM and NN crawlers have significantly higher overlap on targets when compared with the other two combinations. Yet the overlap between the SVM and NN crawlers is no larger than 50%. This result again highlights the observation that the crawlers are investigating different subspaces while retrieving equally valuable sets of relevant pages. This observation is not peculiar to topical crawlers. Katzer et al. [1982] have previously noted that information retrieval systems with similar performances showed low overlap in retrieved sets of documents. For the Web, Lawrence and Giles [1998] found that combining results from 6 different search engines can improve the coverage by 3.5 times as compared to a single search engine on average. The limited overlap of crawled pages between similarly performing crawlers could, in part, be due to the nondeterminism in a multithreaded crawling system. However, any practical crawling infrastructure will have some form of multithreading to be sufficiently fast and efficient.

10. CONCLUSION

In summary, we present a systematic evaluation of classifier-based topical crawling algorithms using a collection of more than 100 topics. Our main findings are:

(1) Identification of better classifiers within each scheme included in the study. We compare the performance of crawlers that use classifiers based on various versions of Naive Bayes, Neural Network, and SVM classification schemes.
(2) The SVM and Neural Network-based classifiers perform equally well for topical crawling. However, the former takes less time to train. Both the SVM and NN crawlers outperform the NB crawler. Thus there is merit in exploring classifier schemes other that NB, which is the one most focused upon in the previous literature.
(3) Reducing the seed set size does not effect the order of performances between the SVM, NN, and NB crawlers. It does, however, change the significance of performance differences between them.
(4) An attempt with adaptation using pseudo examples when the number of original examples are few (5 positive, 10 negative) does not help.
(5) The weakness of the NB crawler was a result of its weak classification accuracy combined with extremely skewed scores that did not allow for effective prioritization over the frontier.
(6) We observe that the similar performing NN and SVM crawlers have no more than 50% average overlap on both the URLs crawled and the targets fetched. Hence, the choice of crawler will have an effect on the end application even if the collections created seem equally good.

An aspect of our current work is that we constrain the context of hyperlinks to be the entire parent page. It may be the case that instead, a context consisting of a subset of terms within the page is more effective. More granular definitions of contexts (words in anchor text or its neighborhood) have been shown to improve crawler performance in previous work [Hersovici et al. 1998; Pant and Menczer 2003]. In parallel research we are investigating various definitions of contexts of hyperlinks and their effect on classifier-based topical crawling.

We note that our results provide us with insights into crawler performances for crawls of $0 - 10,000$ pages. While crawls of up to $10,000$ pages for a given topic or a theme may be sufficient for many applications in research and for personalized agents [Pant and Menczer 2002; Pant et al. 2004b], it remains to be seen if we can generalize the results to crawls of millions of pages. Nevertheless, based on the number of topics, variety of classifiers, and the number of pages crawled per topic, our study is the most extensive in the current topical crawling literature.

The classifiers that guide the crawlers are trained on an ad-hoc proportion (priors) of negative (2/3) and positive (1/3) examples. One cannot assume these to be the real priors over the Web. Moreover, the priors during a crawl can be expected to change with time. Since our seeds are relevant (positive) pages, we may expect the crawler to encounter more positive pages early in the crawl as compared to later. It would be worthwhile to explore techniques that compensate for this drift in the environment during a crawl.

Ensembles of classifiers have been shown to outperform individual classifiers on accuracy [Dietterich 1998; Schapire 1999]. An ensemble combines the

decisions of several classifiers. This could be done in several ways, some as simple as a majority vote. It would be interesting to test if ensemble methods help to improve crawling performance. Given the fact that even similar performing classifier-guided crawlers have low overlap in terms of subspaces covered, we may expect an ensemble-guided crawler to work well. Observe that our evaluation classifiers are used as an ensemble to judge the relevance of the crawled pages.

We can easily extend our experiments with many other classification algorithms (Weka provides implementations for several others such as Linear Regression, Bayes Network, and C4.5). However, any such extension comes at the cost of crawling millions of additional pages spread over a test bed of 100 topics. Another simple extension can be achieved by combining the scores for a given URL if the URL appears on several downloaded pages. At present, we only keep the score from the first page in which the URL was found. As future work, we may apply different heuristics for combining the scores and compare the affects of such heuristics on crawling performance.

Our experiment with retraining the classifier by adding pseudo-examples after crawling for a short span of time did not show merit. Can the quality of pseudo-examples be improved if we emphasize training examples that are authoritative or hub pages? Here, the crawled pages will be picked as pseudo-examples, not just based on the content but also the linkages. This is another direction for future work.

Active efforts at studying special challenges when retrieving information from the Web may be observed. A leading venue for such research is that of the TREC Web track [Craswell et al. 2003]. However, such efforts are yet to translate into a standardized framework for performance evaluation of Web crawlers [Srinivasan et al. 2003; Srinivasan et al. 2005].

In this article, we used previously untested crawling configurations and discovered insights into the comparative performances of different topical crawlers that are guided by classifiers. The entire study was done with an emphasis on systematic, extensive, and statistically robust design and experimentation that involves several crawlers, millions of pages and over more than one hundred topics.

ACKNOWLEDGMENTS

We would like to thank Filippo Menczer, Nick Street, and Shannon Bradshaw for insightful discussions. We also thank the anonymous reviewers for their helpful comments and suggestions.

REFERENCES

AGGARWAL, C. C., AL-GARAWI, F., AND YU, P. S. 2001. Intelligent crawling on the World Wide Web with arbitrary predicates. In *Proceedings of the 10th International World Wide Web Conference*. Hong Kong.

BEN-SHAUL, I., HERSCOVICI, M., JACOVI, M., MAAREK, Y. S., PELLEG, D., SHTALHAIM, M., SOROKA, V., AND UR, S. 1999. Adding support for dynamic and focused search with fetuccino. *Computer Networks and ISDN Systems 31*, 11–16, 1653–1665.

BURGES, C. J. C. 1998. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery 2*, 2, 121–167.

CHAKRABARTI, S., PUNERA, K., AND SUBRAMANYAM, M. 2002. Accelerated focused crawling through online relevance feedback. In *Proceedings of the 11th International World Wide Web Conference*. Hawaii.

CHAKRABARTI, S., VAN DEN BERG, M., AND DOM, B. 1999. Focused crawling: A new approach to topic-specific Web resource discovery. In *Proceedings of the 8th International World Wide Web Conference*.

CHAU, M., ZENG, D., AND CHEN, H. 2001. Personalized spiders for web search and analysis. In *Proceedings of the 1st ACM/IEEE-CS Joint Conference on Digital libraries*.

CHEN, H., CHAU, M., AND ZENG, D. 2002. Ci spider: A tool for competitive intelligence on the Web. *Decision Support Systems* 1–17.

CHEN, H., CHUNG, Y., RAMSEY, M., AND YANG, C. 1998. A smart itsy bitsy spider for the Web. *J. Ameri. Soc. Info. Sci. 49*, 7, 604–618.

CHOW, C. K. 1957. An optimum character recognition system using decision functions. *IRE Transactions* 247–254.

CRASWELL, N., HAWKING, D., WILKINSON, R., AND WU, M. 2003. Overview of the trec-2003 web track. In *Proceedings of TREC-2003*.

CRISTIANINI, N. AND SCHÖLKOPF, B. 2002. Support vector machines and kernel methods: the new generation of learning machines. *AI Magazine 23*, 3, 31–41.

DAVISON, B. D. 2000. Topical locality in the web. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

DAY, M. 2003. Collecting and preserving the World Wide Web. Tech. rep., UKOLN, University of Bath. February. http://library.wellcome.ac.uk/assets/WTL039229.pdf.

DE BRA, P. M. E. AND POST, R. D. J. 1994. Information retrieval in the World Wide Web: Making client-based searching feasible. In *Proceedings of the 1st International World Wide Web Conference* (Geneva).

DIETTERICH, T. G. 1998. Machine-learning research: Four current directions. *The AI Magazine 18*, 4, 97–136.

DILIGENTI, M., COETZEE, F., LAWRENCE, S., GILES, C. L., AND GORI, M. 2000. Focused crawling using context graphs. In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB 2000)*. Cairo, Egypt, 527–534.

DUDA, R. O., HART, P. E., AND STORK, D. G. 2000. *Pattern Classification (2nd Edition)*. Wiley-Interscience.

DUMAIS, S. T. 1998. Using svms for text categorization. *IEEE Intelligent Systems Magazine 13*, 4.

ELKAN, C. 1997. Boosting and naive bayesian learning. In *International Conference on Knowledge Discovery in Databases*.

HERSOVICI, M., JACOVI, M., MAAREK, Y. S., PELLEG, D., SHTALHAIM, M., AND UR, S. 1998. The shark-search algorithm—An application: Tailored Web site mapping. In *Proceedings of the 7th International World Wide Web Conference*.

HOGG, R. V., CRAIG, A., AND MCKEAN, J. W. 2004. *Introduction to Mathematical Statistics*, 6 ed. Prentice Hall.

HORNIK, K., STINCHCOMBE, M., AND WHITE, H. 1989. Multilayer feedforward networks are universal approximators. *Neural Network 2*, 5, 359–366.

JAIN, A. K., MAO, J., AND MOHIUDDIN, K. M. 1996. Artificial neural networks: A tutorial. *Computer 29*, 3, 31–44.

JOACHIMS, T. 2002. Learning to classify text using support vector machines. Ph.D. thesis, Kluwer.

JOHN, G. H. AND LANGLEY, P. 1995. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the 11th Annual Conference on Uncertainty in Artificial Intelligence (UAI 95)*. Montreal, Quebec, Canada, 338–345.

JOHNSON, J., TSIOUTSIOULIKLIS, K., AND GILES, C. L. 2003. Evolving strategies for focused web crawling. In *Proceedings of the 20th International Conference on Machine Learning (ICML-2003)*. Washington DC.

KATZER, J., MCGILL, M. J., TESSIER, J. A., FRAKES, W., AND DAS-GUPTA, P. 1982. A study of the overlap among document representations. *Information Technology: Research and Development 2*, 261–274.

LAWRENCE, S. AND GILES, C. L. 1998. Searching the World Wide Web. *Science 280*, 98–100.

LECUN, Y. 1986. Learning processes in an asymmetric threshold network. In *Disordered Systems and Biological Organization*, E. Bienenstock, F. Fogelman-Soulié, and G. Weisbuch, Eds. Springer-Verlag, Les Houches, France, 233–240.

LEWIS, D. D. 1998. Naive (bayes) at forty: The independence assumption in information retrieval. In *Proceedings of the 10th European Conference on Machine Learning*. Springer-Verlag, 4–15.

LIPPMANN, R. P. 1988. An introduction to computing with neural nets. In *Artificial Neural Networks: Theoretical Concepts*. V. Vemuri Ed. IEEE Computer Society Press, Los Alamitos, CA, 36–54.

MAAREK, Y. S., JACOVI, M., SHTALHAIM, M., UR, S., ZERNIK, D., AND BEN-SHAUL, I. Z. 1997. Webcutter: a system for dynamic and tailorable site mapping. *Computer Networks and ISDN Systems 29*, 8–13, 1269–1279.

MCCALLUM, A. AND NIGAM, K. 1998. A comparison of event models for naive bayes text classification. In *Proceedings of the AAAI 98 Workshop on Learning for Text Categorization*.

MCCALLUM, A. K., NIGAM, K., RENNIE, J., AND SEYMORE, K. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval 3*, 2, 127–163.

MCCULLOCH, W. S. AND PITTS, W. 1943. A logical calculus of ideas imminent in nervous activity. *Bulletin of Mathematical Biophysics 5*, 115–133.

MCLACHLAN, G. J. 1992. *Discriminant Analysis and Statistical Pattern Recognition*. Wiley, New York.

MENCZER, F. AND BELEW, R. K. 2000. Adaptive retrieval agents: Internalizing local context and scaling up to the Web. *Machine Learning 39*, 2–3, 203–242.

MENCZER, F., PANT, G., RUIZ, M., AND SRINIVASAN, P. 2001. Evaluating topic-driven Web crawlers. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

MENCZER, F., PANT, G., AND SRINIVASAN, P. 2004. Topical Web crawlers: Evaluating adaptive algorithms. *ACM Trans. Int. Tech. 4*, 4, 378–419.

MITCHELL, T. M. 1997. *Machine Learning*. McGraw-Hill, New York.

MÜLLER, K.-R., MIKA, S., RÄTSCH, G., TSUDA, K., AND SCHÖLKOPF, B. 2001. An introduction to kernel-based learning algorithms. *IEEE Neural Networks 12*, 2, 181–201.

PANT, G. AND MENCZER, F. 2002. MySpiders: Evolve your own intelligent Web crawlers. *Autonomous Agents and Multi-Agent Systems 5*, 2, 221–229.

PANT, G. AND MENCZER, F. 2003. Topical crawling for business intelligence. In *Proceedings of the 7th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2003)*. Trondheim, Norway.

PANT, G., SRINIVASAN, P., AND MENCZER, F. 2002. Exploration versus exploitation in topic driven crawlers. In *Proceedings of the 11th World Wide Web Workshop on Web Dynamics*.

PANT, G., SRINIVASAN, P., AND MENCZER, F. 2004a. *Web Dynamics*. Springer-Verlag, Chapter Crawling the Web.

PANT, G., TSIOUTSIOULIKLIS, K., JOHNSON, J., AND GILES, C. L. 2004b. Panorama: Extending digital libraries with topical crawlers. In *Proceedings of the 4th ACM/IEEE-CS Joint Conference on Digital Libraries*. 142–150.

PLATT, J. C. 1999. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods Support Vector Learning*, B. Schölkopf and A. Smola, Eds. M.I.T. Press, 185–208.

PORTER, M. 1980. An algorithm for suffix stripping. *Program 14*, 3, 130–137.

QIN, J., ZHOU, Y., AND CHAU, M. 2004. Building domain-specific web collections for scientific digital libraries: A meta-search enhanced focused crawling method. In *Proceedings of the 4th ACM/IEEE-CS Joint Conference on Digital Libraries*.

RENNIE, J. AND MCCALLUM, A. K. 1999. Using reinforcement learning to spider the Web efficiently. In *Proceedings of the 16th International Conference on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 335–343.

RUMELHART, D. E., HINTON, G. E., AND WILLIAMS, R. J. 1986. Learning internal representations by error propagation. *Parallel Data Processing 1*, 318–362.

RUMELHART, D. E., WIDROW, B., AND LEHR, M. A. 1994. The basic ideas in neural networks. *Comm. ACM 37*, 3, 87–92.

SALTON, G. 1971. *The SMART Retrieval System—Experiments in automatic document processing*. Prentice Hall Inc., Englewood Cliffs, NJ.

SALTON, G. AND MCGILL, M. J. 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill.

SCHAPIRE, R. E. 1999. A brief introduction to boosting. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 1401–1406.

SCHÖLKOPF, B., BURGES, C. J. C., AND SMOLA, A. J. 1999. *Advances in Kernel Methods: Support Vector Learning*. MIT Press.

SCHÖLKOPF, B. AND SMOLA, A. J. 2003. A short introduction to learning with kernels. In *Advanced Lectures on Machine Learning*, S. Mendelson and A. J. Smola, Eds. *Lecture Notes in Artificial Intelligence*. Springer-Verlag, New York, NY, 41–64.

SRINIVASAN, P., MENCZER, F., AND PANT, G. 2003. Defining evaluation methodologies for topical crawlers. In *SIGIR 2003 Workshop on Defining Evaluation Methodologies for Terabyte-Scale Collections*. http://dollar.biz.uiowa.edu/~gpant/Papers/crawl_framework_position.pdf.

SRINIVASAN, P., MENCZER, F., AND PANT, G. 2005. A general evaluation framework for topical crawlers. *Information Retrieval 8*, 3, 417–447.

THEODORIDIS, S. AND KOUTROUMBAS, K. 2003. *Pattern Recognition*. Academic Press, San Diego, CA.

VAPNIK, V. N. 1995. *The nature of statistical learning theory*. Springer-Verlag New York, Inc.

WIDROW, B. 1990. 30 years of adaptive neural networks: Perceptron, madaline, and backpropagation. *Proceedings of the IEEE 78*, 9, 1415–1452.

WRIGHT, S. AND NOCEDAL, J. 1999. *Numerical Optimization*. Springer.

YANG, Y. AND PEDERSEN, J. O. 1997. A comparative study on feature selection in text categorization. In *Proceedings of the 14th International Conference on Machine Learning (ICML-97)*. Morgan Kaufmann Publishers, 412–420.

Received July 2004; revised March 2005; accepted May 2005