Enhanced hypertext categorization using hyperlinks

Soumen Chakrabarti
soumen@almaden.ibm.com
IBM Almaden

Byron Dom
dom@almaden.ibm.com
IBM Almaden

Piotr Indyk*
indyk@cs.stanford.edu
Stanford University

Abstract

A major challenge in indexing unstructured hypertext databases is to automatically extract meta-data that enables structured search using topic taxonomies, circumvents keyword ambiguity, and improves the quality of search and profile-based routing and filtering. Therefore, an accurate classifier is an essential component of a hypertext database. Hyperlinks pose new problems not addressed in the extensive text classification literature. Links clearly contain high-quality semantic clues that are lost upon a purely term-based classifier, but exploiting link information is non-trivial because it is noisy. Naive use of terms in the link neighborhood of a document can even *degrade* accuracy. Our contribution is to propose robust statistical models and a *relaxation labeling* technique for better classification by exploiting link information in a small neighborhood around documents. Our technique also adapts gracefully to the fraction of neighboring documents having known topics. We experimented with pre-classified samples from Yahoo![1] and the US Patent Database[2]. In previous work, we developed a text classifier that misclassified only 13% of the documents in the well-known Reuters benchmark; this was comparable to the best results ever obtained. This classifier misclassified 36% of the patents, indicating that classifying hypertext can be more difficult than classifying text. Naively using terms in neighboring documents increased error to 38%; our hypertext classifier reduced it to 21%. Results with the Yahoo! sample were more dramatic: the text classifier showed 68% error, whereas our hypertext classifier reduced this to only 21%.

1 Introduction

Automatic identification of topics for unstructured documents, also called *supervised classification* or *supervised categorization*, is an important operation for text databases. Topic identifiers constitute structured meta-data that can be used to index a text database. Such structuring helps circumvent keyword ambiguity and improves the quality of ad-hoc searching and browsing [4, 5, 12, 16] as well as profile-based routing of documents as in the so-called "push" or filtering services.

Topic identification is one example of extracting structured information from a semi-structured or unstructured source. This is becoming increasingly important as the web is expanding, search engines are proliferating, and text and hypertext are becoming standard data types supported by most modern databases and extenders. Apart from answering specific queries, which we call the *retrieval* problem, many text databases provide support for unsupervised and supervised categorization. The former we call *clustering*; the latter we call *classification*. In this paper we deal exclusively with the classification problem. A classifier is first provided a topic set (which may be flat or hierarchical) with sample *training* documents for each topic. Using these, the classifier *learns* the topics. Later, presented with new documents, it finds the best matching topics.

Text classification has been extensively researched in IR, but large-scale experiments have been largely restricted to homogeneous corpora, e.g., a set of financial articles from newspapers and magazines (e.g., TREC[3] and Reuters[4]), medical documents (MEDLINE[5]), etc. The vocabulary is coherent and the quality of authorship is high. For the Reuters dataset, classifiers based on rule induction or feature selection classify 80% to 87% of the documents correctly [2, 4, 38].

The problem: Hypertext in general and the Web in particular encourage diverse authorship, navigational and citation links, and short, fragmented documents whose topics can be determined only in the broader context of the local region of the link graph. Similar comments are valid for email and newsgroup articles, since references to earlier emails or postings can be considered as citations. The issue of classification based on non-local features (i.e. attributes not directly included in the document being classified) also looms large in other hyperlinked corpora, such as patents and academic publications and their citation graphs.

Hypertext poses new challenges to automatic classifiers. This was evident from our early experiments (described later in detail). The same classifier that was 87% accurate for Reuters, tested on a sample of patents, correctly classified only 64% [4]. On a sample from Yahoo! the performance was even poorer: only 32% of the documents were classified correctly. This is comparable to a recent study by Sahami on a similar sample from Yahoo using a more detailed learning program that used Bayesian nets [18, 30].

The text classifier performed poorly because web documents are extremely diverse, featuring home pages, topical resource lists, hierarchical HTML documents generated automatically from LaTeX, active pages with scripts and links, etc. Even within a broad area there is little consistency in vocabulary. Many web documents are very short, serving merely as a list of resource links. In the patent database, patents in the same class may have diverse authorship across time and assignees.

Our contribution: This paper explores new ways in which information latent in hyperlinks can be exposed to a suitably designed classifier, so that the judgment of topic is based on information that is both local and non-local to the document. The end result is a hypertext classifier that

*The work was performed at IBM Almaden.
[1] http://www.yahoo.com
[2] http://www.ibm.com/patents
[3] http://trec.nist.gov/data/docs_eng.html
[4] http://www.research.att.com/~lewis
[5] ftp://medir.ohsu.edu/pub/ohsumed

Appears in ACM SIGMOD 1998, Seattle, Washington.

achieves significantly improved accuracy at a moderate computational overhead.

This is the first topic classification system known to us that combines textual and linkage features into a general statistical model to infer the topic of interlinked documents. (In §5, we will comment on the superficial similarity with a large body of work on hypertext *retrieval* [8, 14, 22, 36] and discuss how both our problem and technique are essentially different.)

Our exploration starts with an obvious idea for exploiting links commonly found in the aforesaid retrieval literature: including the text of a document's *neighbors* (documents that cite or are cited by it) into it as if they were local terms and then running a classifier on this resulting pseudo-document. In our experiments with patents and Yahoo!, absorbing neighbor text performs *worse* than classification based on local terms alone. One reason is that links from many pages point to pages from a diverse set of topics. There is significant linkage between related topics among patents, e.g. patents on voltage regulators refer frequently to those on electrical transmission, patents on frequency modulators cite those on oscillators, etc. Worse scenarios are seen on the web, often with completely unrelated topics, e.g., web pages of extremely diverse topics link to Netscape or "Free Speech Online."

Thus, link information is noisy, and consequently, the naive approach from retrieval research does not work reliably. Analysis of the failures leads us to a more general model for thinking about non-local features. It becomes clear that the *topics* of neighboring documents, and not necessarily their *terms*, determine linking behavior. Therefore, rather than directly use terms from the neighborhood, we develop a more elaborate model of hypertext in which the topic of a document determines (in a noisy way) the text in it as well as its propensity to link to documents from a set of related topics.

If the topics, or classes, of all neighbors of the test document (that which is being classified) are known, this gives a simple classifier that is more accurate than text-based classifiers. The more likely situation is that only a fraction of the documents in the test document's neighborhood are pre-classified. Our model above leads to a circularity in that case. Fortunately, a wealth of literature in *Markov Random Field Theory* [17, 28, 9] enables us to design an iterative algorithm that initially guesses the classes based on text alone, then updates them iteratively. The class assignment is guaranteed to converge if initiated sufficiently close to a "consistent" configuration.

Small neighborhoods, such as of radius 1–2 links around the test document, appear adequate for significant improvements beyond text-based classifiers. For the patent corpus (which is already authored consistently) the new method cuts down classification error from 36% to 21%, a reduction of 42%. The improvement for the Yahoo! corpus is more dramatic: text-based classification errs about 68% of the time, whereas our method has only 21% error, a reduction of 70%. Combined with the fact that there are over 2 million patents and 100 million web pages, our work represents a significant step forward in robustness. Our method also shows a graceful variation of accuracy as the fraction of documents whose classes are pre-specified changes. The relaxation scheme enhances accuracy even when *no* document in the neighborhood has a pre-specified class!

Organization of the paper: In §2 we start with the basics of fast text classification, data structures, algorithms, and design decisions. Then we introduce the complications of hyperlinks, and show how simplistic ideas of dealing with these complications fail. §3 and §4 are the key sections where we develop our models and algorithms. Experimental evaluation is included in these sections to present a progression of ideas and resulting improvements. We review related work in §5 and conclude in §6.

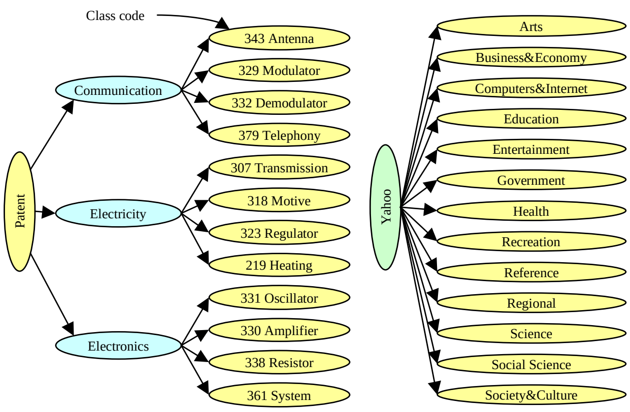


Figure 1: Taxonomies used in our experiments.

2 Hypertext classification basics

In traditional text classification, the object being classified is a self-contained *document*. A document is a sequence of *terms*. Here and elsewhere we will ignore sectioning and formatting cues. During training, documents are supplied to the classifier with attached pre-assigned classes. The classifier develops models for the classes, a process also called *learning*. During testing, the classifier must assign classes to previously unseen documents.

This section has three parts. In the first part we describe our data sets. In the second part we will review pure text-based classification. We will describe the basic classification engine, the statistics that are collected, and how these are used to infer topics for new documents. In the third part we will discuss the essential complication that hypertext brings into the previous setup. A hypertext document is not self-contained. It has *in-neighbors* (other documents citing it) and *out-neighbors* (documents it cites). These, and their neighbors in turn, and so on, form a hypertext graph. For both training and testing, this graph is the view that the classifier has of the corpus. This is quite different from existing literature on classifying structured relational records with self-contained attributes. Assuming an ideal classifier, we can propose an elaborate adaptation of the term-based model to the hypertext setting, but it becomes clear that in practice, this will not be feasible. The rest of the paper develops an alternative approach that is successful.

2.1 Datasets for evaluation

We used two manually pre-classified hyperlinked corpora for our experiments. The first is a sample from IBM's Patent Server database. Our topic hierarchy has 3 first level nodes and 12 leaves, shown in Figure 1 (a). Compared to Reuters, these topics were chosen to pose a more difficult topic recognition problem: there is a large vocabulary overlap and cross-linkage between the topics. For each leaf, 630 randomly selected documents were chosen for training and 300 for testing. Our second corpus is a sample of about 20,000 documents taken from Yahoo! At the time of sampling, the top level of Yahoo! had 13 classes, shown in Figure 1 (b).

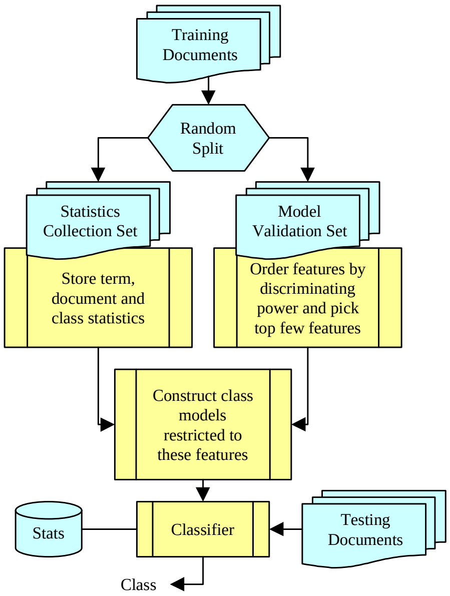


Figure 2: The basic classification engine TAPER at the core of the new classifier that uses both links and terms. A term is any unique ID, so we can freely substitute words, phrases, or class names for a term. We can even run two copies of TAPER, one using only text, the other using only links, and combine the information.

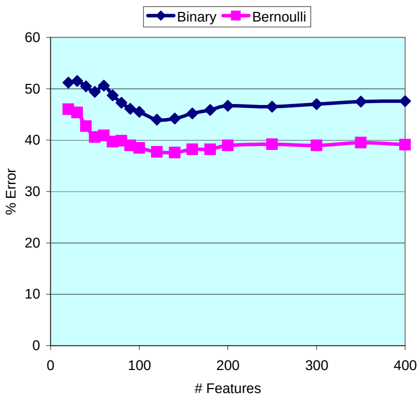


Figure 3: TAPER constructs, for all internal nodes c_0 in the topic taxonomy, a graph as shown above in one pass over the part of the training corpus set aside for model validation. The curve plots error rate against the size of F, the chosen feature set. It typically decreases as useful features are brought into F, and then increases as F starts to include noise. F should be set at the minimum error point. The Bernoulli model performed better than the binary model in our experiments on a text-only classifier. Bernoulli has a lower error rate and less sensitivity to picking too many features.

For later experiments based only on hyperlinks, a subset about 900 of the 20,000 documents was used.

2.2 Text classification

All of the link-based classification techniques will be built on top of a core classifier called TAPER[6] that we have built. TAPER was used for text classification, but is built so that the notion of a document is very general: it is a multiset of arbitrary tokens. TAPER learns a topic tree from sample training documents. Documents belonging to multiple topics are logically copied into each topic. Given a new document, the goal is to find the best matching topics from the tree, starting at the root. In this process, each internal node represents a decision point where a classifier specific to that internal node matches the document with the immediate children of that node. TAPER constructs a diverse set of classifiers specifically suited to each internal node. A sketch of the system is shown in Figure 2. Here we will briefly discuss the feature selection and the class models.

Training starts with scanning the documents and collecting term statistics. We reiterate, in anticipation of the following sections, that as far as TAPER is concerned, a term is a 32-bit ID, which could represent a word, a phrase, words from a linked document, etc. This and later phases of TAPER are heavily optimized to handle up to 2^{32} tokens and documents and 2^{16} topic nodes. For lack of space, such details of system design and implementation are omitted; they can be found in our earlier work [4].

The next step after scanning is called *feature selection*. Fix an internal node c_0 and its children c, c_1, c_2, etc., and consider the classifier at c_0, whose goal is to route documents into those subtrees that best match a test document. Some terms are better than others for this purpose because they occur significantly more frequently in a few classes compared to the others. These terms are *good discriminators*; the others are *noise*, and should be discarded before the classifier forms document models for the classes. Note that these good discriminators can be diverse for different internal nodes.

Finding the best discriminators depends on the statistical model one assumes the data is drawn from. Many proposed models for text, related to Poisson and multinomial processes, exist in the literature. Unfortunately, for most such models, finding exactly the optimal subset of terms out of a lexicon of 50,000–100,000 terms appears impractical. In our system, rather than search for arbitrary subsets, we first *order* the terms by decreasing ability to separate the classes; one notion of such ability that we have derived from the Pattern Recognition literature [10] and found effective, is the following score:

$$\mathrm{score}(t) = \frac{\sum_{c_1,c_2}\big(\mu(c_1,t)-\mu(c_2,t)\big)^2}{\sum_c \frac{1}{|c|}\sum_{d\in c}\big(f(t,d,c)-\mu(c,t)\big)^2}, \qquad (1)$$

where c, c', c_1, c_2 are children of internal node c_0, $f(t,d,c)$ is the number of times term t occurs in document d in the training set of class c, with document length normalized to 1, and $\mu(c,t) = \frac{1}{|c|}\sum_{d\in c} f(t,d,c)$. We compute the score of all terms, order the terms by decreasing score, and pick a set F of terms that is a *prefix* of the above sequence that gives the best classification accuracy over a (random) validation set (Figures 2 and 3).

After feature selection, a classifier is associated with each c_0. Suppose c_0 has children $c_1, \ldots, c_\ell$. Given a class model, the classifier at c_0 estimates model parameters for each child. When a new document is input, the classifier evaluates, using the class models and Bayes's law, the posteriori probability of the document being generated from each child $c \in \{c_1, \ldots, c_\ell\}$ in turn, given it was generated from c_0. We use very simple document models for large scale computational efficiency. In the *Bernoulli* model, every class c has an associated coin with as many sides as there are terms in the lexicon. Each face of the coin corresponds to some term t and has some success probability $\theta(c,t)$, estimated using $f(t,d,c)$ and additional statistics collected during the document scanning step. A document in the class is gener-

[6]TAPER stands for Taxonomy And Path Enhanced Retrieval.

ated by repeatedly flipping the coin and writing down the term corresponding to the face that turns up. Assuming this model, we have:

$$\Pr[d \in c | c_0, F] \;=\; \frac{\pi(c) \prod_{t \in d \cap F} \theta(c,t)^{n(d,t)}}{\sum_{c'} \pi(c') \prod_{t \in d \cap F} \theta(c',t)^{n(d,t)}}, \quad (2)$$

where $\theta(c,t)$ is the probability that "face" t turns up when "coin" c is tossed. $\pi(c)$ is the *prior* probability of class c, typically, the fraction of documents in the training or testing set that are from class c. $n(d,t)$ is the number of times term t occurred in document d. The details of the estimation of θ have been described elsewhere [4]. An alternative *binary* model truncates $n(d,t)$, the number of times term t occurs in document d, to a $\{0,1\}$ value [18]. The estimation of θ changes slightly, but much of the above framework remains unchanged. Classification over the entire taxonomy can then be posed as a shortest path problem on the taxonomy. We omit the details.

Figure 3 compares classification error against $|F|$, the size of the feature set, for binary and Bernoulli models for a node in the US Patent topic tree; similar results are seen for Reuters newswire and Web documents. Based on these observations we fix the core classifier to use the Bernoulli model for the rest of the paper.

2.3 The complications of hypertext

We will now discuss how hyperlinked documents upset the self-contained, clean document models above.

2.3.1 Statistical foundations: the basic model

We begin by laying a statistical foundation for our analysis. We consider the corpus to be a set of documents (also called nodes) $\Delta = \{\delta_i, i = 1, 2, \ldots, n\}$ and these are linked by directed links $i \to j$. Let $G(\Delta)$ be the graph defined by these documents and links, and let $A = A(G)$ be the associated adjacency matrix: $A = \{a_{ij}\}$ where $a_{ij} = 1$ if the link $i \to j$ exists and $a_{ij} = 0$ if it doesn't. Let τ_i represent the text of document i and let $T = \{\tau_i\}$ represent the entire collection of text corresponding to Δ. Each τ_i is a sequence $\{\tau_{ij} | j = 1, \ldots, n_i\}$ of tokens. Tokens include terms, formatting controls, and links. We will ignore formatting controls, and deal with citations separately from terms. For most of the paper, we will also use τ as a multiset rather than a sequence.

When a classifier is being trained, its input is not merely a set of documents, but this graph and an assignment from nodes to class labels. When the classifier has to label a new document, it has as input not only that document, but also some *neighborhood* of that node in the graph, and is free to make use of any information in this neighborhood, perhaps the whole corpus.

We want to classify the documents Δ into a set of classes or categories $\Gamma = \{\gamma_i, i = 1, 2, \ldots, m\}$. Let c_i be the class assignment of document i and let $C = \{c_i\}$ the set of class assignments for the entire collection Δ. Assuming that there is a probability distribution for such collections, we want to choose C such that $\Pr(C|G,T)$ is maximum. This constitutes a *Bayes Classifier*, which gives the minimum expected error rate over all classifiers [10]. $\Pr(C|G,T)$ can be decomposed using Bayes rule:

$$\Pr[C|G,T] \;=\; \frac{\Pr[C,G,T]}{\Pr[G,T]} \;=\; \frac{\Pr[G,T|C]\ \Pr[C]}{\Pr[G,T]}, \quad (3)$$

$$\text{where} \quad \Pr[G,T] \;=\; \textstyle\sum_{C'} \Pr[G,T|C']\ \Pr[C'].$$

Because $\Pr(G,T)$ is not a function of C, it will suffice to choose C to maximize $\Pr(G,T|C)\ \Pr(C)$. In some cases the categories of some of the documents will be known. To indicate this we divide C into two disjoint sets: C_K (the known classes) and C_U (the unknown classes). In these terms the classification problem is that of selecting C_U to maximize $\Pr(C_U|C_K,G,T)$. In many cases (e.g. where the document collection is the entire web) we only care about a subset of the classes in C_U.

In order to perform the estimation of C_U we need a form for $\Pr(C,G,T)$. It is extremely unlikely that we can obtain a known function for this. Thus, we will have to estimate it. In order to do that we must assume some functional form that will have parameters that must be estimated. (We should point out that even so-called "non-parametric" forms have this property.) In reality, of course, it will be impractical to attempt to estimate a form for $\Pr(C_U|C_K,G,T)$ where $\{C_K,G,T\}$ appear as a simple list of all the associated values (e.g. links and terms). A big part of the challenge in finding such a form is to find a set of *features* that extract in a condensed form that information most useful for performing categorization. In addition to the definition and selection of features another device for simplifying the computation and estimation associated with $\Pr(C_U|C_K,G,T)$ is to make certain independence assumptions. These are frequently not correct (as will be case here) but making them can be seen to constitute a kind of approximation. Ultimately, of course, their validity will be judged by the accuracy of the associated classification obtained.

In this general framework all the information contained in C_K, G and T is potentially useful in classifying a single document δ_i. Making use of all this information, however, is both computationally infeasible and of questionable value. The strong intuition one has is that the relevance of various aspects (classes, links and terms) of these have falls off rapidly with distance (number of links) from δ_i.

2.3.2 Using text from neighbors

With the above discussion in mind, we begin with an examination of the usefulness of terms from neighboring documents. Consider a hypertext document δ_i. (For the moment we consider a document to be a multiset, not a sequence, of terms.) We can transform δ_i into a super-document δ_i' that includes not only the terms in δ_i but all terms in the hypertext graph, in a specially tagged fashion (otherwise all documents would be identical). Thus, `cat` in δ_i remains `cat` in δ_i'. If δ_i points to δ_j, which has term `cat` in it, the special term `O@cat` is added to δ_i'. The "O" signifies an out-link. Similarly, the term `cat` appearing in a page pointing to δ_i is added to δ_i' as `I@cat`. If δ_j points to δ_i and δ_k, where δ_k contains term `cat`, the special term `IO@cat` appears in δ_i. Prefixes like `OIO@` are *tags* to separate the original terms from these engineered features. This seems preferable to merging both local and non-local terms into the same space, as done in earlier retrieval research [8, 14, 22, 36].

In qualitative terms, it is conceivable that a classifier loses no necessary information when presented with δ_i' (which is now gigantic) rather than δ_i and its associated corpus graph. We can let a feature selection process automatically decide the importance of all these features, and throw out the noise. Given an adequate volume of training data (and the time to make use of it) the worst that the inclusion of these additional terms can do is to have no effect on classification accuracy, but we might expect it to result in some improvement. Intuitively, we would expect that terms in nearby neighbors will be used by the classifier, whereas terms that are far away in the graph will be discarded as noise. In practical situations, however, this may not necessarily be the case. This can be seen as a signal-to-noise issue;

there may be so many irrelevant terms that fitting-the-noise effects could become inevitable with realistic training corpus sizes. We next experiment on these initial ideas.

3 The radius-one specialization

Our experimental setup will have the following flavor. During training, each sample document will have a pre-assigned class, and the classifier will be permitted to explore any neighborhood of the sample and know the assigned classes of any document in the neighborhood. During testing, the test document's class will be hidden. Furthermore, we will control and vary the fraction of documents in the neighborhood whose pre-assigned classes are "exposed" to the classifier on demand. The exposed documents will be chosen uniformly at random from the neighborhood that the classifier chooses to explore to classify the test document.

During testing, the algorithms will have the following general form. First, the classifier will grow a neighborhood graph around the test document. It will then use features directly, or derived from, the neighborhood to classify the test document. In the case where all immediate neighbors of the test document are exposed, this completes the algorithm (§3.1 and §3.3). The case where only a fraction of the neighborhood is exposed is dealt with in §3.4. We design an algorithm that bootstraps off a text-only classifier, and then iteratively updates the classes of documents in the neighborhood until stability is achieved. This update is done by a classifier that combines local and non-local information. In this paper, in the interest of computational feasibility, we restrict non-local features to within a radius of one or two. A typical academic paper, a patent in the US Patent Database, and an average web page all have typically more than ten citations or out-links. Thus a radius-one neighborhood already has the potential to slow down a classifier by a significant factor, unless the classifier is engineered carefully. Note that the radius of the initial *neighborhood* is unrelated to, and can be larger than, the "radius of influence" discussed above.

3.1 Experiments with text from neighbors

Perhaps the most obvious approach to implementing a "radius of influence" of one is to use terms from the immediate neighbors of the test document as if they were in the test document itself. In view of the discussion in §2, we propose the following alternatives. We run the experiments on the patent corpus, starting with the most favorable setting where all neighbors of training and test documents have known classes.

Local: For both training and testing, a pure term-based classifier was used on the text of each patent itself. Thus, no use is made of hyperlinks. This is the baseline.

Local+Nbr: For both training and testing, the complete text of all its neighbors (patents that cite it as well as patents it cites) is first concatenated with the text of each patent. The result is input to a pure term-based classifier.

Local+TagNbr: Same as above, but the terms absorbed from neighbors are distinguished from original local terms by tags, so that the classifier can distinguish between the signal content of local vs. non-local terms.

Figure 4 shows the results. Two disappointing observations emerge. First, the error rate *increases* from 36% to 38.3% when neighbor text is used, in our experiments. Second, tagging shows very small improvement. We can provide the following explanations and comments after careful scrutiny of the results. Although these observations are for our particular corpus, the problems may arise in other hyperlinked corpora as well, e.g., Salton and Zhang found that inclusion of cited titles can add spurious words that degrade retrieval quality [33] (also see §5).

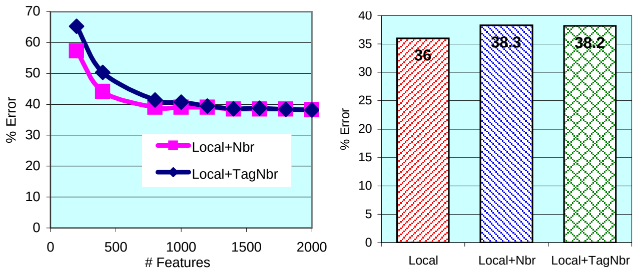


Figure 4: Result of classification using local and neighbors' terms, tagged and untagged. For the patent corpus, neighbor text is worse than local text, and tagging does not help.

Why did neighbor text do worse? Even after feature selection, there was too much of it (an average patent has over 5–10 neighbors), and their term distribution was not sufficiently similar to the distribution of the "true" class of the patent (which has to happen if this trick is to work well). We found frequent cross-linkage between different classes in patents and certainly the web ("this site is best viewed using..."); also see the discussion in §1. Including terms from such neighbors might easily lead to complete misclassification.

Why did tagging not help? Tagging splits a term into many forms and can make it relatively rare, even if the corresponding untagged term is not as rare. Dimensionality of text is already uncomfortably large compared to typical training set sizes. We further challenged the classifier with many more features but not more documents. Although an ideal classifier should do no worse on `Local+TagNbr` compared to `Local`, our heuristic feature selection and approximate class models got overwhelmed by the signal-to-noise ratio.

Are there simple fixes to these problems? We seem to run into a *catch-22* situation on both counts. Consider first the problem of the original document δ_i getting swamped with terms from "large" neighbors. One may propose various term weighting schemes; e.g. include terms in δ_i itself each with weight 1, terms in radius-one neighbors each with weight $\frac{1}{2}$, terms in radius-two neighbors with weight $\frac{1}{4}$, and so on. At what rate should the weights decay? Depending on this rate, spurious terms may be included, leading to misclassification (if the non-local terms have high weight) or informative remote terms may be neglected.

Next, consider the "noisy" neighbor scenario. It is clear that there are at least some links that lead to useful neighbors, but how can we identify these? A simple heuristic is to restrict the expansion to out-links within the site or domain. This will miss valuable information in at least two forms: a highly informative link from the Database group at the University of Wisconsin to IBM Almaden will be ignored, as will be the observation that many Stanford students link to both UW and IBM Database groups, thus inducing an implicit topical similarity between them.

3.2 Feature engineering

It is clear from the above discussion that it will be impractical and not very useful to use all the text in T (the whole corpus) to classify a single document. At the other extreme, most text-classification algorithms/systems use only the text τ_i contained in the document δ_i. For corpora that are not hyperlinked, this is all that is available to use in performing the classification. We may find it useful to use more that just the text τ_i to classify δ_i, however, and we hope to find some "happy medium" between these two extremes. How can we do this while favoring our classifier with a good signal-to-noise ratio?

A similar problem appears in inductive logic programming [23]. Suppose we are given a relational table, or several tables, containing information about patients with hypertension. Apart from local attributes of a person such as age and weight, there can be other attributes such as father and mother, whose records we shall assume are also in the database. Assuming there is a relation between lineage and hypertension, we wish to use the latter fields for inducing rules to classify patients as having high or low risk of heart attack. To do this, we can augment each record with fields from records of the parents. Rather than using raw data from the related records, one can pre-process those raw attribute values into a synthesized feature. One example is whether the parent was classified as high or low risk. This is called *feature engineering* [1].

3.3 Using class information from pre-classified neighbors

Noting that text from neighbors is too noisy to help classification, we reflect on the process and rationale for citation in the first place. In this section, we explore the following thesis: if the classes for all documents neighboring δ_i were known, replacing each hyperlink in δ_i with the *class ID* of the corresponding document, could provide a highly distilled and adequate (for classification) representation of δ_i's neighborhood.

In order to make use of the neighbor class labels in this way, we need an appropriate *classifier*. The general/ideal form for this classifier is that shown in equation (3). As we mentioned previously, we can drop $\Pr(G, T)$ and simply choose (in principle) C to maximize $\Pr(G, T|C)\Pr(C) = \Pr(G, T, C)$. This is fine if we know the functions $\Pr(G, T|C)$ and $\Pr(C)$ and we have a computationally feasible procedure for finding the global maximum of their product as a function of C. Even if we don't know these functions *a priori*, if we know their functional/parametric forms and have a suitable learning procedure (to estimate the parameter values) and a feasible optimization procedure, we can still achieve this. The usual situation is significantly less ideal however. We must frequently "guess" the functional forms, selecting those that (1) we believe to have sufficient degrees of freedom to describe the necessary inter-class decision boundaries; and (2) for which we have feasible learning (estimation) and optimization algorithms. In a specific problem it will not, in general, be possible to satisfy both of these simultaneously. This may be due to the models and algorithms at our disposal or the more fundamental problem of "intrinsic confusion" where the classes overlap in the feature space corresponding to the features we have chosen for performing classification.

First, we consider the problem of classifying a single document δ_i, for which we assume that we know all the classes of all its immediate neighbors. To perform this classification we will use the usual Bayes classifier. That is, we choose c_i to maximize $\Pr(c_i|\mathcal{N}_i)$, where, in this context, $\mathcal{N}_i$ represents the collection of all the known class labels of all the neighbor documents. This collection can be further decomposed into in-neighbors and out-neighbors: $\mathcal{N}_i = \{\mathcal{I}_i, \mathcal{O}_i\}$. Manipulation of this using Bayes' Law, as was done in equation (3) for C, leaves us with the problem of choosing c_i to maximize:

$$\Pr(\mathcal{N}_i|c_i)\ \Pr(c_i) \tag{4}$$

Our estimate for $\Pr(c_i)$ is simply the frequency of the class c_i in the training corpus. For $\Pr(\mathcal{N}_i|c_i)$ we will assume independence of all the neighbor classes. That is:

$$\begin{aligned}&\Pr(\mathcal{N}_i|c_i) \\ &\quad = \textstyle\prod_{\delta_j\in\mathcal{I}_i}\Pr(c_j|c_i, j\to i)\prod_{\delta_k\in\mathcal{O}_i}\Pr(c_k|c_i, i\to j).\end{aligned} \tag{5}$$

We will use the following notation to represent the terms in these products:

$$\begin{aligned}\pi(c_i) &\equiv \Pr(c_i),\\ \phi(c_j, c_i|\mathcal{I}) &\equiv \Pr(c_j|c_i, j\to i),\\ \phi(c_j, c_i|\mathcal{O}) &\equiv \Pr(c_j|c_i, j\to i),\\ n(\gamma_j, i|\mathcal{I}) &\equiv \#\text{in-neighbors of } \delta_i \text{ labeled } \gamma_j\\ n(\gamma_j, i|\mathcal{O}) &\equiv \#\text{out-neighbors of } \delta_i \text{ labeled } \gamma_j\end{aligned}$$

Using this notation and rewriting equation (5) as a product over classes rather than neighbors yields the following form for equation (4):

$$\begin{aligned}&\Pr(\mathcal{N}_i|c_i)\Pr(c_i) \\ &\quad = \pi(c_i)\textstyle\prod_{j=1}^{m}\left[\phi(\gamma_j, c_i|\mathcal{I})\right]^{n(\gamma_j, i|\mathcal{I})}\left[\phi(\gamma_j, c_i|\mathcal{O})\right]^{n(\gamma_j, i|\mathcal{O})}.\end{aligned} \tag{6}$$

3.4 Iterative relaxation labeling of hypertext graphs

In §3.1 we investigated the problem of classifying a hypertext document using both its own text and that of its neighbors. Then, in §3.3, rather than using individual terms from neighbors, we investigated the problem of using the class labels of preclassified neighbors. In this section, we examine using all three (document text, neighbor text and neighbor classes), but in the more realistic setting wherein only some or none of the neighbor classes are known. In those cases where the neighbor classes are known *a priori* we will use those class labels as the sole feature representation of the associated documents. Where the neighbor classes are not known, on the other hand, we will indirectly use the text and link structure of the entire collection Δ in a type of *relaxation labeling* scheme.

Before we consider this complete problem, we discuss the combined use of known neighbor classes and terms from only the document to be classified. We assume that there is no direct coupling between the text of a document and the classes of its neighbors. We assume only an indirect coupling through the class of the document itself. This assumption can be expressed precisely using conditional probabilities:

$$\Pr[\mathcal{N}_i, \tau_i|c_i] \quad = \quad \Pr[\mathcal{N}_i|c_i]\ \Pr[\tau_i|c_i]. \tag{7}$$

Following the discussion of previous sections, the resulting classifier can be written as:

$$c_i \quad = \quad \arg\max_{\gamma}\{\Pr[\mathcal{N}_i|\gamma]\ \Pr[\tau_i|\gamma]\ \Pr[\gamma]\}, \tag{8}$$

where the Bernoulli forms for the components of this equation can be obtained from equations (2), (6) and γ is a dummy class-label variable for c_i.

If all the classes of the neighbors are known *a priori*, this is the classifier we would use. If, as is usually the case, some or all of the neighbor classes are unknown, how do we proceed? We do not wish to ignore the unclassified pages, so we need a bootstrap mechanism: we first classify the unclassified documents from the neighborhood (for example by using a terms-only classifier) and then use this knowledge to classify the given document, in the same way as described above. This procedure can be performed iteratively, to increase the quality of the classification. This iteration constitutes a kind a *relaxation labeling*, a technique that has been applied in the computer vision and image processing fields [17, 28].

Let Δ_K symbolize everything that is *known* about Δ. Using our previous notation, $\Delta_K = \{G, T, C_K\}$. Using this we want to estimate the unknown classes C_U. Let Δ_U represent the corresponding documents (i.e. those whose classes are unknown). If we are seeking to classify a single document δ_i, we would like to choose c_i to maximize $\Pr(c_i|\Delta_K)$. Note that, letting $\mathcal{N}_i^U \equiv \mathcal{N}_i \cap \Delta_U$, this can be expressed as:

$$\begin{aligned} \Pr(c_i|\Delta_K) &= \textstyle\sum_{\mathcal{N}_i^U \in \Omega_i} \Pr(c_i, \mathcal{N}_i^U|\Delta_K) \qquad (9) \\ &= \textstyle\sum_{\mathcal{N}_i^U \in \Omega_i} \Pr(c_i|\mathcal{N}_i^U, \Delta_K) \Pr(\mathcal{N}_i^U|\Delta_K), \end{aligned}$$

where Ω_i is the set of all possible class labelings of the documents in $\mathcal{N}_i$. There are m^k of these where k is the number of documents in $\mathcal{N}_i$. To make this formula more manageable we make the following approximations/assumptions:

1. Limited range of influence:
$$\Pr(c_i|\mathcal{N}_i^U, \Delta_K) = \Pr(c_i|\mathcal{N}_i^U, \mathcal{N}_i^K),$$
where $\mathcal{N}_i^K = \mathcal{N}_i \cap \Delta_K$. This is equivalent to assuming a first-order *Markov Random Field* [6].

2. Independence among the neighbor class probabilities:
$$\Pr(\mathcal{N}_i^U|\Delta_K) = \textstyle\prod_{\delta_j \in \mathcal{N}_i^U} \Pr(c_j|\Delta_K).$$

(It is desirable to remove the latter assumption via a detailed experimental study of neighbor class distributions.) We observe that with these approximations the collection of all such equations (9) constitute a coupled system of equations in which the variables are the probability estimates: $\{\Pr(c_i|\Delta_K), i = 1, 2, \ldots, n\}$, which suggests the following iterative solution:

$$\begin{aligned} &\Pr(c_i|\Delta_K)^{(r+1)} \qquad (10) \\ &= \textstyle\sum_{\mathcal{N}_i^U \in \Omega_i} \left[\Pr(c_i|\mathcal{N}_i^U, \mathcal{N}_i^K) \prod_{\delta_j \in \mathcal{N}_i^U} \Pr(c_j|\Delta_K)^{(r)} \right], \end{aligned}$$

where r is the iteration index.

This relaxation is guaranteed to converge to a locally consistent assignment provided it is initiated "close enough" to such a consistent state [28].

Note that Δ_K includes all the text T, the link structure G and the known classes C_K and that $\mathcal{N}_i^K$ includes the corresponding subsets of these. The way in which this information is applied in this estimation problem is controlled by the function $\Pr(c_j|\mathcal{N}_i^U, \mathcal{N}_i^K)$. In the work described here we use the Bernoulli forms of equations (2) and (6). A pseudocode sketch follows:

```
Given test node δ0
Construct a radius-r subgraph Gr(δ0) around δ0
Assign initial classes to all δ ∈ Gr(δ0) using local text
Iterate until consistent:
  Recompute the class for each δ based on
    local text and class of neighbors
```

Some further approximations are made to make evaluation of the sum in equation (10) tractable. These will be described in the sequel. A final note regarding equation (9). It was written in terms of only the immediate neighbors $\mathcal{N}_i = \{\mathcal{N}_i^U, \mathcal{N}_i^K\}$, but it can be extended to larger neighborhoods, i.e., link distance two and beyond. This would constitute a higher-order Markov Random Field assumption.

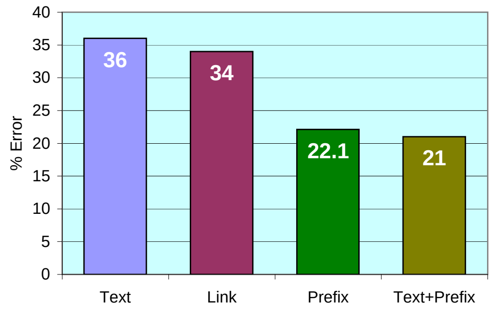


Figure 5: Comparison of error rates for Text, Link, Prefix and Text+Prefix.

3.5 Experiments and performance

We complete this section on unit "radius of influence" with a series of experiments on patents. The first set will deal with the case where the classifier can query the "true" class of any document except for the one that is being classified. The goal will be to design a classifier that can use both local text and links to immediate neighbors to advantage. We call this the *completely supervised* scenario. In the second part, we will assume the more realistic scenario in which the true classes of only a fraction of the documents in the vicinity of the test document are known to the classifier. This will involve the relaxation scheme described earlier, and will be called the *partially supervised* scenario.

3.5.1 The completely supervised case

We compare the following options using the patent corpus.

Text: Only local text was used for training and testing.

Link: The only document features are the class names of neighboring patents. In the Patent Server (as well as, say, Yahoo), class names are *paths* in a topic hierarchy, looking like this:

```
[29][METAL WORKING]
  [X][PROJECTILE MAKING]
    [Y][.Bullet or shot]
      [Z][..Jacketed or composite]
```

(Here the first [] contains the numeric class code and the second [] contains a descriptive string.) Our features are the full path names, e.g. /29/X/Y/Z. Note that these paths can be longer and therefore more focused than our 2-level dataset.

Prefix: We include all *prefixes* of the class paths of neighbors as features, i.e. all of /29, /29/X, /29/X/Y, and /29/X/Y/Z. The reason for this will become clear in a moment.

Text+Prefix: We run two copies of TAPER, one that only uses local text, and another that uses only the class prefixes as above. Then the probabilities are combined as if the joint distribution of term and link frequency is product of their marginal distributions. This gives a hybrid classifier.

The results are shown in Figure 5. The text baseline, as before, is 36% error. Using links in the form of full paths gives negligible benefit; the error is 34%. However, a dramatic effect is seen with Prefix: the error drops to 22.1%. The combined Text+Prefix model improves on this slightly to give a final error rate of only 21%.

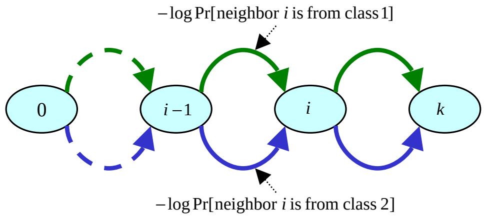


Figure 6: Exponentially many paths determine one neighbor configuration each, but we can prune the sum over configurations to only include those that have the highest probability. These correspond to the shortest $0 \to k$ paths in the graph.

This is an example where the feature selection mechanism of TAPER is exploited well. Link performs worse than expected because the features (full class paths) are too detailed and occur too rarely to become strong signals. However, when we expose all the prefixes to TAPER, it is free to discard paths that are too short (almost equally frequent in all documents) or too long (too infrequent to be statistically significant).

3.5.2 The partially supervised case

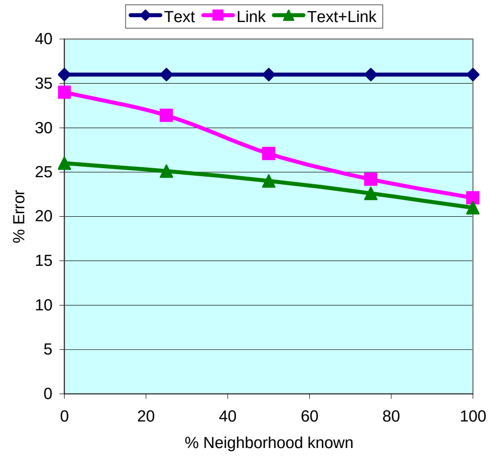
Finally we study the realistic scenario in which only a *fraction* of the documents in the vicinity of a test document have classes known to the classifier. The foundations of the iterative relaxation-based classification has already been laid in §3.4. Here we add some comments about implementation.

Each of these neighbors of δ_i can potentially belong to any class. Thus there is a large number of possible assignments of the neighbors to classes. If there are k neighbors and $|\Gamma|$ classes, the neighbors could be in any of $|\Gamma|^k$ configurations, and that many terms in the sum shown in equation (10). As per earlier discussion in the completely supervised case, we assume that the probability of a fixed class assignment $\mathcal{N}_i^U$ is the product of probabilities of the appropriate class assignment of each neighbor of δ_i. Similarly, the completely supervised case gives us a recipe to estimate $\Pr[c_i|\mathcal{N}_i^U, \mathcal{N}_i^K]$, once the configuration of $\mathcal{N}_i^U$ is fixed. The only problem is that the sum has an enormous number of terms.

Luckily, most of the configurations are highly improbable. Consider the graph in Figure 6. For simplicity, let there be $|\Gamma| = 2$ classes. There are nodes numbered $0, \ldots, k$, and two edges from node $i-1$ to i. One edge corresponds to the case where the i-th neighbor belongs to class γ_1, the other, for class γ_2. Suppose we annotated these edges with edge weights which are the negative of the logarithm of the probability of the respective class assignment. Then the shortest path from 0 to k corresponds to the largest probability configuration.

Observe that although there are 2^k neighbor configurations, it is easy to find the highest probability configuration in time $O(k \log k + k|\Gamma|)$ time via a shortest path computation. It also turns out that we can extract the shortest P paths in time $O(k|\Gamma| + k \log k + P \log P)$ [11]. We have not observed any adverse effects of such truncation of (10) on the accuracy of classification. Typically, after the top two or three class choices, the remaining classes have probabilities as low as 10^{-30} and can be ignored. On the other hand, the large range of numbers encountered during the computation requires some care with the floating point computations so as not to lose precision.

Figure 7(a) shows the results of the iterative relaxation



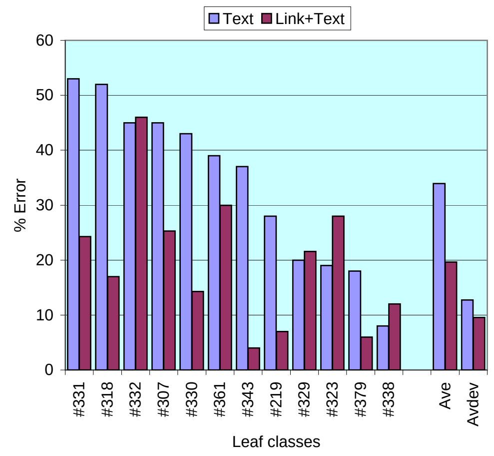


Figure 7: (a) Results of relaxation labeling based hypertext classification. The x-axis is the percentage of documents in the neighborhood of the test document whose true classes are revealed to the classifier on demand; these are chosen uniformly at random from the neighborhood. The y-axis if the error rate for three schemes. One uses local text only, and is thus not really a relaxation scheme. The others use links, and both links and text, in the iterative step. Both are "seeded" by a text-only classifier. (b) A class-by-class break-down of the differences in accuracy between Text and Link+Text.

labeling-based classifier. In the graph, the x-axis is the percentage of documents in the neighborhood of the test document with known classes. The y-axis if the error rate for three schemes. Since the text-based classifier does not use links, its error rate is fixed. There are a number of interesting points to note about the other lines in the graph. Obviously, adding link information is significantly boosting accuracy, cutting down error by up to 42%. More importantly, the benefits are seen even when only a relatively small fraction of the neighborhood has known classes, and there is a graceful variation in the overall accuracy as this fraction varies. The text-based and text-and-link-based classifiers use of the order of 50,000 features. In contrast, the link-based classifier uses only 15 (for this taxonomy). Thus it has a tiny footprint and is very fast. However, it does well only when a reasonably large fraction of the neighborhood has known classes. Text+Link always beats Link by some margin, but this is quite small at the completely supervised end of the spectrum. Text+Link is also stabilized by

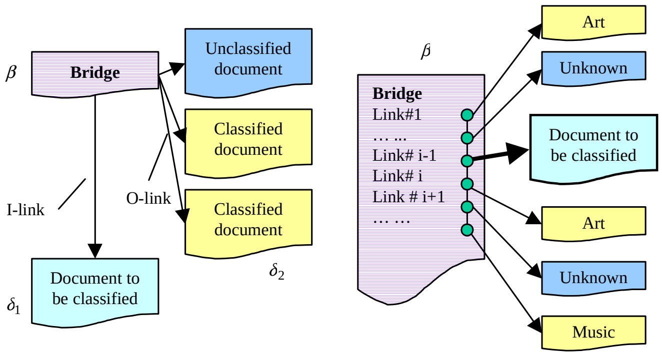


Figure 8: The phenomenon of pages that *bridge* topics across length-two paths. The second diagrams shows how one may try to exploit locality in the topics on bridge pages.

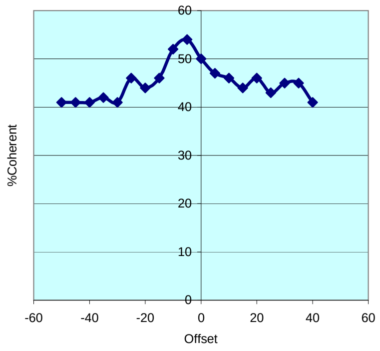


Figure 9: Is there locality of topic in out-link sequences? Plotted above is the percentage of cases for which outlinks at a fixed offset from a given outlink point to some document of the same topic as the given document at offset 0. There appears to be a positive "background" probability that even long list of outlinks are coherent, and the bridge page contains links mostly to one topic. But there are also pages where the topic of outlinks drifts from topic to topic; this is revealed by the noticeable peak near zero. It is interesting that the peak is not *at* zero. The explanation we can offer is that the outlink to the most popular pages occur later in a document.

text at the extreme where the neighborhood is completely unknown. It is the more stable relaxation method, but is computationally somewhat more expensive.

What is most intriguing is that even in the *most unfavorable* setting of zero knowledge about the neighborhood, using only a link-based iteration scheme, and seeding it only once using local terms, a small but visible increase in accuracy is seen (between Text and Link at 0%). This gain is entirely from implicit constraints imposed on the probability models by the link structure.

Also shown in Figure 7(b) is a break-up of the performance over the 12 different leaf patent classes. Perhaps it should come as no surprise that the improvement is diverse; in cases it is negative. This depends on the linkage behavior between different classes. It is interesting that apart from greatly reducing average error across classes, the relaxation scheme also reduces (by about 26%) the average *deviation* of error across the classes.

4 The radius-two specialization

In a homogeneous corpus such as the Patent database, it is meaningful to assign each patent to a class, and to estimate parameters like the expected fraction of outlinks from documents in one class that point to documents belonging to another class. In contrast, the Web is so diverse that no topic taxonomy can hope to capture all topics in sufficient detail. There will always be documents, in the vicinity of the documents of interest to us or to the classifier, that cannot be meaningfully classified into the "known universe." A cursory measurement shows that only about 19% of Yahoo! documents have an in-link from some other Yahoo! document. Only 28% have an out-link to some Yahoo! document and about 40% have some link with another Yahoo! page.

Exploring larger neighborhoods on the Web can be futile and dangerous. The benefits of finding a few known neighbors may be offset by noise, or worse, strong but incorrect signals collected in the process. For example, a large fraction of web pages point to popular sites like Netscape or Alta Vista, even though the topic of these sites may be completely unrelated. This may not be fixed by a static list of "stopword" sites, since different sites may assume this role for different topics.

4.1 Bridges

Co-citation is a well-studied phenomenon in linked corpora, such as academic papers [32]. Documents that cite or are cited by many common documents may be regarded as similar, much as documents sharing many terms are adjudged similar. Citation-based similarity, or a combination of citation and term-based similarity, can then be used to perform *unsupervised* clustering of documents [40].

These common documents hint that two or more pages have the same class, while not committing what that class could be. We call these documents *bridges*. Figure 8(a) illustrates the scenario. Two documents δ_1 and δ_2 are said to be *bridge-connected* if there exists a document β pointing to both δ_1 and δ_2; β is the bridge. To go from δ_1 to δ_2 one traverses edge (β, δ_1) against its direction and then (β, δ_2). We call this an *IO-path* because we follow an Inlink to β followed by an Outlink. We call β an *IO-bridge* for δ_1 and δ_2.

Thus there are four ways to go to a neighbor two links away. The choice of IO-paths, out of II, OO, IO, and OI, is guided by domain knowledge. The difference between I and II, or O and OO, is one of degree. Of IO and OI, we claim that IO is more meaningful. (Consider perhaps some 80% of the Web today OI-bridged to each other because they all point to Netscape.) Fortunately, IO-bridges abound on the web. In fact, every topical "resource page" is a potential bridge. The Yahoo! and Infoseek sites thus include thousands of bridges on all sorts of topics. The results of the following experiment hints that bridges can be a powerful support for link-based classification.

Experiment: For each page δ in our Yahoo! sample, we consider all pages $\delta_1 \ldots \delta_k$ pointing to it. For our purpose, we regard each page δ_i as an ordered list O_i containing the out-links of δ_i. Some of the out-links in O_i point to pages not known to Yahoo!, but some others are. For each out-link in O_i which pointed to a web page $\delta' \neq \delta$ contained in Yahoo, we checked if the class of δ and δ' are the same; if so we call the pair (δ, δ') *coherent*. Denote $(\delta' - \delta)_{O_i}$ to be the difference between the positions of δ and δ' in O_i. This is negative if the out-link to δ' occurs before that to δ and vice-versa. Finally, for each integer offset D we computed a fraction of coherent pairs among all pairs (δ', δ) for which $(\delta' - \delta)_{O_i} = D$ for some i. (See Figure 8(b).)

The results are shown on Figure 9. The graph plots an estimate of the probability of staying on the same topic as

δ as we move off before and after the link to δ in some page pointing to δ. Two aspects of the graph are worth noting. First, the probability appears not to vanish to zero as we move farther away from the link to δ; this means that many long bridges are almost *pure*: they point to pages from the same Yahoo! topic. The second is that there *is* a peak near zero, which implies that outlinks near the one to δ are more likely to be the same topic as δ. We discuss how to exploit the former property in the rest of this section; in the next we deal with impure bridges.

TAPER with IO-bridges: In our general framework, a neighborhood is grown around the test document and then classes are assigned iteratively to each document in a manner influenced by other documents in its "radius of influence." In §3, this radius was at most one. Here we will use IO-bridges to exert influence at radius two. Our experimental setup is as follows.

1. We consider a sample of documents from Yahoo! such that each is IO-bridged to at least one other page in Yahoo! (In this experiment we do not consider normal terms as features at all, so we need pages that have links to them.) We randomly divide this set into 70% for training and 30% for testing. This gave us about 594 training and about 255 testing documents. Whereas this is not nearly enough for text-based classification, our small feature space made this small dataset usable.

2. For training, we use the following features from each document δ. We look at all the documents in the training or testing set that are IO-bridged to the document under scrutiny. These have known class paths in topic taxonomy. We take all prefixes of these paths and construct an engineered document δ'.

3. For testing, we use the class paths for all documents in the *training* set that are IO-bridged, but not the test set.

4. These features are input to TAPER as usual.

5. We compare the above scheme with a purely term-based classification of the same test set using TAPER, and the same training set as above.

The results are shown in Figure 10 (the set of middle bars). Owing to a variety of factors (fewer terms per document, diverse authorship, unused hyperlinks) the pure term-based classifier performs rather poorly, having an error rate of 68%. Furthermore, this error rate was achieved at a large feature set size of over 14,000 terms. It appears that this may be a basic limitation of text-based classification; a more detailed learning program that used Bayesian nets showed comparable performance with a comparable sample from Yahoo! [30]. In contrast, the IO-bridge based classifier has excellent performance: it uses only 14 features (the class labels) and has only 25% error!

4.2 Locality in bridges

Sometimes an IO-bridge may not be "pure"; it may include a sequence of topics, with many out-links for each topic. Thus we may need to segment such a bridge into segments, each of which points to pages with a coherent topic. E.g., a personal hotlist may have a list of Physics resources followed by a list of Math resources.

Segmenting a web page into intervals corresponding to coherent topics appears to be a difficult task. In Machine Learning, algorithms have been discovered for solving related but simpler problems, such as segmenting a sequence of coin toss outcomes (using a number of hidden coins) such that each segment is likely to be generated by a single coin [15]. These algorithms already have high complexity which makes them inapplicable in our context.

Therefore, we resort to following *approximate* approach. We define two sets of features: one which is guaranteed to contain all bridge information but can contain additional noise, and the second which consists almost exclusively of bridge information but is likely to be incomplete. The former option was already explored in the previous section, where we depended on TAPER to bring out the signal despite the noise. Here we will explore the latter option.

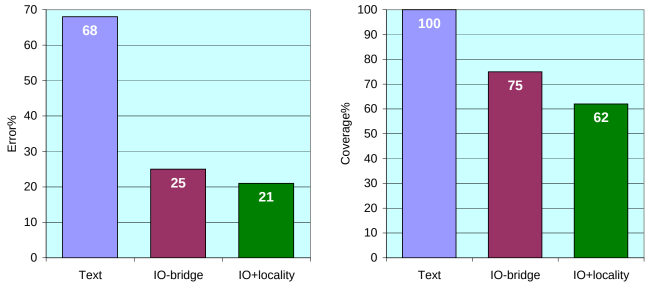


Figure 10: Comparison of a pure term-based classifier with two versions of a pure link-based classifier that uses IO-bridges. The link-based classifiers uses orders of magnitude fewer features (14 as against 14,000). The first one uses IO-bridges; it reduces error rate from 68% to 25%. The second one exploits locality in IO-bridges; it reduces error rate further to only 21%. This happens at some cost of "coverage": whereas a pure text-based classification will accept any document and assign it some class, the two link-based classifiers will have enough information to draw a conclusion respectively 75% and 62% of the time.

IO-bridges with locality: More specifically, a class ID c is included as a "feature" for page δ if the following holds: There exists an IO-bridge β pointing to δ with three outlinks $\delta_{\text{before}}, \delta, \delta_{\text{after}}$ in that order, where the classes of δ_{before} and δ_{after} are known and equal, and no out-links between them point to a document with a known class. In other words, we take all pages β pointing to δ and check if the classes of the closest classified pages before and after the link to δ equal; if so, we include that class as a feature in the engineered page δ'.

Observe that here we are trading coverage for precision here. Some useful features at radius two may be lost, but a feature that makes it into δ' is very likely to be very valuable. Figure 9 indicates that with probability at least 0.4, links out of IO-bridges go to a single topic; therefore, a feature in δ' can be noise only if the topic changed at least twice between δ_{before} and δ_{after} is adequately small.

Having defined this new engineered feature set, we evaluated their quality on the set of 849 Yahoo pages described earlier. The classification error rate was only 21% (of the pages that had non-empty feature sets under the above feature engineering rule). This is shown in Figure 10. Here, again, the number of features used was 14, against the 14,000 of the term-based classifier.

As mentioned before, the dramatic boost in accuracy is at some cost of coverage: some documents end up having no features at all under the somewhat stringent feature engineering rule. Whereas the coverage of pure text-based classification is 100%, that of IO-bridges is 75%, and that of IO-bridges with locality is 62%. This is also shown in

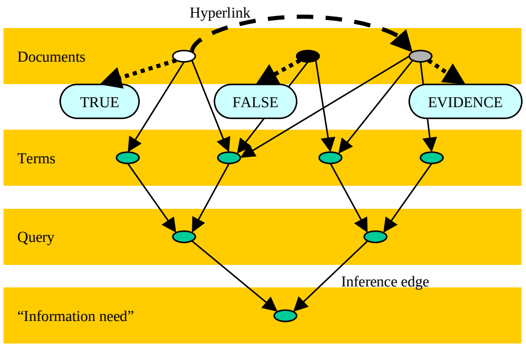


Figure 11: Relevance ranking using inference networks.

Figure 10. This is consistent with our rationale behind the choice of these features: IO-bridges with locality is of higher quality than IO-bridges; however the latter contains more information and therefore can be used to classify more pages.

5 Related work

We discuss three areas of research related to our work: text and hypertext information retrieval, machine learning in contexts other than text or hypertext, and computer vision and pattern recognition.

5.1 Hypertext and information retrieval

Hypertext analysis has been extensively researched long before the Web existed. Starting in 1975, Kwok, in a series of papers [19, 20, 21], proposed a measure of document similarity based on the degree of term intersection with titles of cited documents. Using a small set of 37 medical abstracts, Kwok showed that using citation titles leads to good cluster separation. Althought these are the papers most related to our work, they do not deal with statistical procedures for feature selection or hierarchical classification based on a model of hyperlinks as we do.

The idea of using non-local terms from linked documents has also been used in a large number of papers on *retrieval*. In 1963, Salton [31] proposed using terms associated by citation for retrieval, but later experiments by Salton and others [22, 33] raised the following issue that we address in this paper:

> Important terms may be supplied in some instances, producing substantial performance improvements; in other cases, the process adds indifferent or poor terms to the content description ... No theory exists which would help in distinguishing valuable term associations from less valuable ones.

Since then, many authors have used variants of this approach on different corpora, with varying success. Croft and Turtle had mild success with the CACM and CISI corpora [8]. With the same corpora, Savoy proposed a scheme in which artificial links are added at query time through relevance feedback; this showed a more significant improvement [34, 35, 36]. Frei and Steiger annotated links with frequent terms from the source and target documents to enhance retrieval in UNIX manuals [13, 14]. Smith and Chang used captions from HTML pages to construct a text index for searching for embedded images on the Web [39].

Index terms collected from documents and their linked neighborhoods have been used in vector-space retrieval systems [31] and systems based on inference networks. We discuss the latter approach in some detail. Lucarella and Zanzi [24] propose a rule-based inference framework based on a set of predicates. Similar relations between queries, terms, and documents are captured by a four-layer directed acyclic Bayesian network used by Croft and Turtle [7, 8], shown in Figure 11. The layers represent documents, terms, a query, and the user's "information need." Some of these relations are shown below:

Croft&Turtle	Lucarella&Zanzi
—	query q is about subject c
`about(node,concept)`	document d is about subject c
`synonym(concept1,concept2)`	subject c_1 is related to subject c_2
`cites(node1,node2)`	—

It is important to distinguish between the physical links represented by `cites` and the edges in the inference network, represented by the above relations. (E.g., on the web, the link graph has cycles, whereas an inference network is necessarily acyclic.) During retrieval, each document node in the inference network is set to `TRUE` in turn and the probability of that document satisfying the information need is evaluated using the inference network. The documents are then presented, sorted by decreasing probability. Citations are handled by introducing "evidence" nodes that reinforce terms in documents linked to the document being evaluated.

Notice that `about(node,concept)` is an *input* to the inference (edges connecting the top two layers); the inference does not influence the strength of this relation. These works do not discuss how to compute the `about` relation; they use simple weighting schemes to estimate how important a term is in a document. No notion of learning associations between topics/concepts and terms is proposed.

These systems retrieve and rank by relevance, in addition to documents that contain query terms, those that are neighbors to such documents. In systems that retrieve documents in response to any ad-hoc query it is not adequate to regard this retrieval problem as a supervised 2-way classification (relevant vs. not relevant). Such systems must be able to respond to queries for which they have never encountered training data.

In contrast, our system learns relations between predefined concepts and observable hypertext features such as terms and local link structure, and is therefore different from the prior art. Also note that our problem required further sophistication compared to non-local term absorption.

5.2 Machine learning and data mining

Decision tree classifiers such as CART [3] are widely used for classifying numerical and categorical data. These have been adapted to scale to large relational databases for data mining purposes [25, 37]. These typically work on a single relational table giving attributes of each entity (e.g., a customer). However, little appears to have been done with relationships between entities. In the case where entities are patients, this could be useful in, say, diagnosing diseases. This is a very difficult problem in general; a few specific situations have been handled in the inductive logic programming literature [29, 26, 27, 23]. However, these techniques have not yet been applied to large-scale data mining scenarios [1]. Another difference is the difference in dimensionality [4]. Decision-tree classifiers handle up to hundreds of features or attributes, whereas text corpora often have a lexicon in the hundred thousands.

5.3 Computer vision and pattern recognition

Some of our methods are inspired by image-analysis literature. Relaxation labeling and Markov Random fields have been used in attacking the problems of edge detection, image restoration (noise removal), image segmentation (grouping pixels into contiguous regions) and region labeling (assigning labels to the regions produced by a segmentation process, frequently in the context of object recognition). Our problem is a generalization of these. In our case there can be many classes, including topic taxonomies; an average document has many more neighbors than have pixels or regions in images; and the classes of documents' neighbors provide information in an indirect way: pixels near edge pixels are more likely to be edge pixels, but patents citing patents on antennas can also be about transmitters. In pattern recognition, pixel classification is studied in a more general context of *relaxation labeling* and the convergence properties of relaxation-labeling algorithms are studied using *Markov random fields* [17, 28, 9].

6 Conclusion

We have developed new methods for automatically classifying hypertext into a given topic hierarchy, using an *iterative relaxation algorithm*. After bootstrapping off a text-based classifier, we use both local text in a document, as well as the distribution of the estimated classes of other documents in its neighborhood, to refine the class distribution of the document being classified. Using even a small neighborhood around the test document significantly boosts classification accuracy, reducing error up to 70% from text-based classifiers. Our method also handles the case where only a fraction of the neighborhood is pre-classified. Even when *no* document in the neighborhood is pre-specified, our method improves on term-based classifiers. In future we plan to explore better estimates of the class-conditional probability distributions of the features and better heuristics to explore larger neighborhood graphs to further improve our accuracy.

Acknowledgments. Thanks to Mark Jackson, Alex Miller, Bruce Hönig, and Carol Thompson for help with IBM Patent Server experiments, to Prabhakar Raghavan for helpful discussions, and to Martin van den Berg for comments on the paper.

References

[1] K. Ali. Learning relational rules from relational data. Data Mining Seminar, IBM Almaden, San Jose, CA, Sept. 1997.

[2] C. Apte, F. Damerau, and S. M. Weiss. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 1994. IBM Research Report RC18879.

[3] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth & Brooks/Cole, 1984. ISBN: 0-534-98054-6.

[4] S. Chakrabarti, B. Dom, R. Agrawal, and P. Raghavan. Using taxonomy, discriminants, and signatures for navigating in text databases. In *VLDB*, Athens, Greece, Aug. 1997. Invited submission to VLDB Journal, 1998.

[5] C. Chekuri, M. Goldwasser, P. Raghavan, and E. Upfal. Web search using automatic classification. In *Sixth World Wide Web Conference*, San Jose, CA, 1996.

[6] R. Chellappa and A. Jain. *Markov Random Fields: Theory and Applications*. Academic Press, 1993.

[7] W. B. Croft and H. R. Turtle. A retrieval model for incorporating hypertext links. In *ACM Hypertext*, pages 213–224. ACM, 1989.

[8] W. B. Croft and H. R. Turtle. Retrieval strategies for hypertext. *Information Processing and Management*, 29(3):313–324, 1993.

[9] W. Deng and S. S. Iyengar. A new probabilistic relaxation scheme and its application to edge detection. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, 18(4):432–437, Apr. 1996.

[10] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.

[11] D. Eppstein. Finding the k shortest paths. In *Symposium on the Foundations of Computer Science*. IEEE, 1994.

[12] D. Florescu, D. Koller, and A. Levy. Using probabilistic information in data integration. In *VLDB*, Athens, Greece, 1997.

[13] H. P. Frei and D. Steiger. Making use of hypertext links when retrieving information. In *ACM European Conference on Hypertext (ECHT)*, pages 102–111. ACM, Nov. 1992.

[14] H. P. Frei and D. Steiger. The use of semantic links in hypertext information retrieval. *Information Processing and Management*, 31(1):1–13, 1995.

[15] Y. Freund and D. Ron. Learning to model sequences generated by switching distributions. In *Proceedings of the Eighth Annual ACM Conference on Computational Learning Theory (COLT)*, pages 41–50, 1995.

[16] M. Hearst and C. Karadi. Cat-a-Cone: An interactive interface for specifying searches and viewing retrieval results using a large category hierarchy. In *Proceedings of ACM SIGIR 97*, pages 246–255. ACM Press, 1997.

[17] R. A. Hummel and S. W. Zucker. On the foundations of relaxation labeling processes. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, PAMI-5(3):267–287, May 1983.

[18] D. Koller and M. Sahami. Hierarchically classifying documents using very few words. In *International Conference on Machine Learning*, volume 14. Morgan-Kaufmann, July 1997. To appear.

[19] K. L. Kwok. The use of titles and cited titles as document representations for automatic classification. *Information Processing and Management*, 11:201–206, 1975.

[20] K. L. Kwok. A document-document similarity measure based on cited titles and probability theory and its application to relevance feedback retrieval. In C. J. van Rijsbergen, editor, *Research and Development in Information Retrieval*, pages 221–232. Cambridge University Press, 1984.

[21] K. L. Kwok. A probabilistic theory of indexing and similarity measure based on cited and citing documents. *Journal of the American Society for Information Science*, 36(342–351), 1985.

[22] K. L. Kwok. On the use of bibliographically related titles for the enhancements of document representations. *Information Processing and Management*, 24(2):123–131, 1988.

[23] N. Lavrac and S. Dzeroski. *Inductive Logic Programming - Techniques and Applications*. Chichester, 1994.

[24] D. Lucarella and A. Zanzi. Information retrieval from hypertext: an approach using plausible inference. *Information Processing and Management*, 29(3):299–312, 1993.

[25] M. Mehta, R. Agrawal, and J. Rissanen. SLIQ: A fast scalable classifier for data mining. In *Proc. of the Fifth Int'l Conference on Extending Database Technology*, Avignon, France, March 1996.

[26] S. Muggleton and C. Feng. Efficient induction of logic programs. In *Proceedings of the Workshop on Algorithmic Learning Theory*. Japanese Society for Artificial Intelligence, 1990.

[27] M. Pazzani, C. Brunk, and G. Silverstein. A knowledge-intensive approach to learning relational concepts. In *Machine Learning: Proceedings of the Eighth International Workshop (ML91)*, Ithaca, NY, 1991. Morgan Kaufmann.

[28] L. Pelkowitz. A continuous relaxation labeling algorithm for markov random fields. *IEEE Transactions on Systems, Man and Cybernetics*, 20(3):709–715, May 1990.

[29] R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5, 3:239–266, 1990.

[30] M. Sahami. Web classification using Bayesian nets. Personal communication, Oct. 1997.

[31] G. Salton. Associative document retrieval techniques using bibliographic information. *Journal of the ACM*, 10(4):440–457, Oct. 1963.

[32] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[33] G. Salton and Y. Zhang. Enhancement of text representations using related document titles. *Information Processing and Management*, 22(5):385–394, 1986.

[34] J. Savoy. A learning scheme for information retrieval in hypertext. *Information Processing and Management*, 30(4):515–533, 1994.

[35] J. Savoy. A new probabilistic scheme for information retrieval in hypertext. *The New Review of Hypermedia and Multimedia*, pages 107–134, 1995.

[36] J. Savoy. An extended vector processing scheme for searching information in hypertext. *Information Processing and Management*, 32(2):155–170, Mar. 1996.

[37] J. Shafer, R. Agrawal, and M. Mehta. SPRINT: A scalable parallel classifier for data mining. In *Proc. of the 22th Int'l Conference on Very Large Databases*, Bombay, India, Sept 1996.

[38] Y. Singer and W. W. Cohen. Context-sensitive learning methods for text categorization. In *SIGIR*. ACM, 1996.

[39] J. R. Smith and S.-F. Chang. Visually searching the web for content. *IEEE Multimedia*, 4(3):12–20, 1997.

[40] R. Weiss, B. Velez, M. A. Sheldon, C. Nemprempre, P. Szilagyi, A. Duda, and D. K. Gifford. HyPursuit: A hierarchical network search engine that exploits content-link hypertext clustering. In *ACM Hypertext*, Washington, DC, Mar. 1996.