Using taxonomy, discriminants, and signatures for navigating in text databases

Soumen Chakrabarti Byron Dom Rakesh Agrawal Prabhakar Raghavan

IBM Almaden Research Center

Abstract

We explore how to organize a text database hierarchically to aid better searching and browsing. We propose to exploit the natural hierarchy of topics, or *taxonomy*, that many corpora, such as internet directories, digital libraries, and patent databases enjoy. In our system, the user navigates through the query response not as a flat unstructured list, but embedded in the familiar taxonomy, and annotated with document signatures computed *dynamically* with respect to where the user is located at any time. We show how to update such databases with new documents with high speed and accuracy. We use techniques from statistical pattern recognition to efficiently separate the *feature* words or *discriminants* from the *noise* words at each node of the taxonomy. Using these, we build a multi-level classifier. At each node, this classifier can ignore the large number of noise words in a document. Thus the classifier has a small model size and is very fast. However, owing to the use of context-sensitive features, the classifier is very accurate. We report on experiences with the Reuters newswire benchmark, the US Patent database, and web document samples from Yahoo!.

1 Introduction

The amount of on-line data in the form of free-format text is growing extremely rapidly. As text repositories grow in number and size and global connectivity improves, there is a pressing need to support efficient and effective information retrieval (IR), search and filtering. A manifestation of this need is the recent proliferation of over one hundred commercial text search engines that crawl and index the web, and several subscription-based information multicast mechanisms. Nevertheless, there is little structure on the overwhelming information content of the web.

Proceedings of the 23rd VLDB Conference
Athens, Greece, 1997

It is common to manage complexity by using hierarchy[1], and text is no exception. Many internet directories, such as Yahoo![2], are organized as hierarchies. IBM's patent database[3] is organized by the US Patent Office's class codes, which form a hierarchy. Digital libraries that mimic hardcopy libraries support some form of subject indexing such as the Library of Congress Catalogue, which is again hierarchical.

We will explore the opportunities and challenges that are posed by such topic hierarchies, also called *taxonomies*. As we shall show, taxonomies provide a means for designing vastly enhanced searching, browsing and filtering systems. They can be used to relieve the user from the burden of sifting specific information from the large and low-quality response of most popular search engines [5, 26]. Querying with respect to a taxonomy is more reliable than depending on presence or absence of specific keywords. By the same token, multicast systems such as PointCast[4] are likely to achieve higher quality by registering a user profile in terms of classes in a taxonomy rather than keywords.

The challenge is to build a system that enables search and navigation in taxonomies. Several requirements must be met. First, apart from keywords, documents loaded into such databases must be indexed on *topic paths* in the taxonomy, for which a *reliable* automatic hierarchical classifier is needed. As one goes deep into a taxonomy, shared jargon makes automatic topic separation difficult. Documents on stock prices and on impressionist art look very different to us, but may be carelessly filed as "human affairs" by a Martian. Second, the taxonomy should be used also to present to the user a series of progressively refined *views* of document collections in response to queries. Third, the system must be *fast*, especially since it will often be used in conjunction with a crawler or newswire service. Fourth, the system must efficiently *update* its knowledge when it makes mistakes and a human intervenes.

We describe such a taxonomy and path enhanced retrieval system called TAPER. For every node in the taxonomy, it separates *feature* and *noise* terms by com-

[1] A hierarchy could be any directed acyclic graph, but in this paper we only deal with trees.
[2] `http://www.yahoo.com`
[3] `http://patent.womplex.ibm.com`
[4] `http://www.pointcast.com`

puting the best *discriminants* for that node. When classifying new documents, only the feature terms are used. Good features are few in number, so the class models are small and the classification is speedy. In contrast to existing classifiers that deal with a flat set of classes, the feature set changes by context as the document proceeds down the taxonomy. This filters out common jargon at each step and boosts accuracy dramatically. Addition and deletion of documents is easily handled and discriminants recomputed efficiently. The text models built at each node also yield a means to summarize a number of documents using a few descriptive keywords, which we call their *signature* (this is unrelated to the features). We report on our experience with TAPER using the Reuters newswire benchmark[5], the US patent database, and samples of web documents from Yahoo!. Depending on the corpus, we can classify 66–87% of the documents correctly, which is comparable to or better than the best known numbers. We can process raw text at over seven megabytes a minute on a 133 MHz RS6000/43P with 128 MB memory.

Organization of the paper. In Section 2 we demonstrate that using a taxonomy, topic paths, and signatures can greatly improve the search experience. Next, in Section 3 we study the problems that must be solved to provide the above functionality. The problems are hierarchical classification, feature selection, and document signature extraction. These are explored in detail in Sections 3.3, 3.4, and 3.5 respectively. The proof of quality of signatures is necessarily anecdotal at this point; some examples can be found in Section 2. More rigorous evaluation of feature selection and classification is presented in Section 4. Related work is reviewed in Section 5 and concluding remarks made in Section 6.

2 Capabilities

Most queries posted to search engines are very short. Such queries routinely suffer from the *abundance* problem: there are many aspects to, and even different interpretations of the keywords typed. Most of these are unlikely to be useful. Consider the wildlife researcher asking AltaVista[6] the query `jaguar speed` [5]. A bewildering variety of responses emerge, spanning the car, the Atari video game, the football team, and a LAN server, in no particular order. The first page about the animal is ranked 183, and is a fable. Thwarted, we try `jaguar speed -car -auto`. The top response goes as follows: "If you own a classic Jaguar, you are no doubt aware how difficult it can be to find certain replacement parts. This is particularly true of gearbox parts." The words `car` and `auto` do not occur on this page. There is no cat in sight for the first 50 pages. We try LiveTopics[7], but at the time of writing, all the clusters are about cars or football. We try again: `jaguar speed +cat`. The top two hits are about the clans Nova Cat and Smoke Jaguar; then there is LMG Enterprises, fine automobiles. All these pages include the term `cat` frequently. The 25th page is the first with information about jaguars, but not exactly what we need. Instead, we can go to Yahoo!, drill down into Science:Biology, and query `jaguar`. This takes us to Science:Biology:Animal_Behavior, but we could not find a suitable page about jaguars there.

2.1 Querying in a taxonomy

Suppose we could somehow unite the coverage of AltaVista with the careful, manually designed topic structure of Yahoo!. The query `jaguar speed` would then elicit not a list of documents, but a list of topic paths:

- Business_and_Economy:Companies:Automotive
- Recreation:
  - Automotive
  - Games:Video_Games
  - Sports:Football
- Science:Biology:Animal_Behavior

The user can now restrict queries by *concept*, not by *keyword*. Using samples, it is possible to show the above response even as the user types the query, *before* actually issuing a search. At this point, the user can restrict the search to only a few topic paths. The artificial limit to the length of the response list from search engines, together with cars and video games, will not crowd out the cat. As we have shown above, enforcing or forbidding additional keywords cannot always be as effective. If new documents can be binned into these topic paths in real-time, this ability may be very useful for multicast channels as well. User profiles will be topic paths rather than keywords.

2.2 Context-sensitive signatures

AltaVista's exhaustive keyword index is perhaps more of a problem than a solution. A single occurrence of a term in a document, no matter how useless an indicator of the contents, is indexed. The IR literature has advanced further; there exist prototypes that extract *signature* terms which are then used for indexing. These signatures can also be used as summaries or thumbnails; their descriptive power can often compare favorably with that of arbitrary sentences as extracted by popular search engines. They are also effective for describing a document cluster [1].

We claim that the common notion of a document abstract or signature as a function of the document

[5] `http://www.research.att.com/~lewis/`

[6] `http://www.altavista.digital.com`

[7] `http://www.altavista.digital.com/av/lt/help.html`

alone is of limited utility. In the case of a taxonomy, we argue that a useful signature is a function of both the document and the reference node; the signature includes terms that are "surprising" *given* the path from the root to the reference node. In the above example, *car* and *auto* may be good signature terms at the top level or even at the Recreation level, but not when the user has drilled down to Recreation:Automotive. Here is another illustration from a document[8] in Health:Nursing that goes like this:

> Beware of the too-good-to-be-true baby that is sleeping and sleeping and doesn't want to nurse. Especially monitor the number of wet diapers, as seriously jaundiced babies are lethargic.

The first level classification is Health. We can compute the top signature terms with respect to Health as:

> Jaundice, dampen, dehydration, lethargic, hydrate, forcibly, caregiver, laxative, disposable.

This tells us the document is about treating jaundice. The second level classification is Health:Nursing. Shifting our reference class, we compute the new signature to be:

> Baby, water, breast-feed, monitor, new-born, hormone.

Now we know the document is about nursing *babies*; this information comes from both the path and the signatures. Later we shall propose some means of computing context-sensitive signatures. Thus, significant improvement in search quality may be possible by maintaining functionally separate indices at each taxonomy node, using only a few signature terms from each document.

Another application of context-sensitive signatures is finding term associations. Using phrases for search and classification can potentially boost accuracy. The usual way to find phrases is to test a set of terms for occurrence rate far above that predicted by assuming independence between terms. Unfortunately, associations that are strong for a section of the corpus may not be strong globally and go unnoticed. E.g., *precision* may be visibly associated with *recall* in a set of documents on IR, but not in a collection also including documents on machine tools. Computing signatures at each node exposes all such associations.

2.3 Context-sensitive feature selection

Separating feature terms from noise terms is central to all of the capabilities we have talked about. In the above examples, *car* and *auto* should be "stopwords" within Recreation:Automotive and hence be pruned from the signatures. Feature and noise terms must be determined at each node in the taxonomy.

It is tricky to hand-craft the stopwords out of domain knowledge of the language; *can* is frequently included in stopword lists, but what about a corpus on waste management? The contents of a stopword list should be highly dependent on the corpus. This issue looms large in searching using categories and clusters. In hierarchical categories, the importance of a search term depends on the position in the hierarchy [26].

We will later design an efficient algorithm to find, for each node in the taxonomy, the terms that are best suited for classifying documents to the next level of the taxonomy. Conversely, we detect the noise words that are of little help to distinguish the documents. We reuse the term "feature-selection" from pattern recognition to describe this operation.

Feature selection enables fine-grained classification on a taxonomy. For diverse top-level topics, a single-step classifier suffices. But as a document is routed deep into a taxonomy, shared jargon makes sophisticated feature selection a necessity. Together with feature selection, we have to pick models for each class and a classifier. Many options have been evaluated [31]. In spite of its simplicity, naive Bayesian classifiers are often almost as accurate as more sophisticated classifiers [18]. For a fixed number of features, naive Bayes is faster than more complex classifiers. However, to approach the latter in accuracy, naive Bayes typically needs many more features.

Finding feature terms for each node mitigates this problem. We shall see later that fewer than 5–10% of the terms in the lexicon suffice to discriminate between documents at any node in the taxonomy. This can greatly speed up classification. Fast multi-level classification is not only a database population issue. With increasing connectivity, it will be inevitable that some searches will go out to remote sites and retrieve results that are too large for direct viewing. There are already several "meta-search" tools that forward queries to a number of search engines and combine the results; we have seen how a hierarchical view is much better.

3 Techniques

In this section we will present in detail the techniques that make possible the capabilities mentioned before.

3.1 Document model

There have been many proposals for statistical models of text generation. One of the earliest indicators of the power of simple statistical tests on term frequencies is Zipf's law [38]. The models most frequently used in the IR community are Poisson and Poisson mixtures [28, 33]. (If X is distributed Poisson with rate μ, denoted $X \sim \mathcal{P}(\mu)$, then $\Pr[X = x] = e^{-\mu}\mu^x/x!$ and if Y is distributed Bernoulli with n trials and mean np, denoted $Y \sim \mathcal{B}(n,p)$, then $\Pr[Y = y] = \binom{n}{y}p^y(1-p)^{n-y}$. As $n \to \infty$ and $p \to 0$, the distribu-

[8] http://www2.best.com/~goodnews/practice/faq.htm

tions $\mathcal{B}(n,p)$ and $\mathcal{P}(np)$ converge to each other.) We will assume a Bernoulli model of document generation for the rest of the paper. In this model, a document d is generated by first picking a class. Each class c has an associated multi-faced coin; each face represents a term t and has some success probability $f(c,t)$. Then a document length $n(d)$ is arbitrarily fixed and each term is generated by flipping the coin. We set up some notation.

$$\begin{aligned}
n(d,t) &= \text{number of occurrences of } t \text{ in } d \\
n(d) &= \text{number of terms in } d \\
x(d,t) &= \tfrac{n(d,t)}{n(d)}, \quad \text{the occurrence rate of } t \text{ in } d \\
n(c,t) &= \textstyle\sum_{d \in c} n(d,t) \\
n(c) &= \textstyle\sum_t n(c,t) \\
f(c,t) &= \text{occurrence rate of } t \text{ in } c \text{ (details later).}
\end{aligned}$$

For the moment we can assume $f(c,t) = n(c,t)/n(c)$; we will modify this definition later. Assuming the Bernoulli model, if document d is from class c, then the face probabilities are $\Pr[t|c] = f(c,t)$. Thus,

$$\Pr[d|c] = \binom{n(d)}{\{n(d,t)\}} \prod_t f(c,t)^{n(d,t)}, \qquad (1)$$

where $\binom{n(d)}{\{n(d,t)\}} = \frac{n(d)!}{n(d,t_1)!\, n(d,t_2)! \cdots}$ is the multinomial coefficient.

We appreciate that the independence assumptions are far from the truth. First, given a term has occurred once in a document it is more likely to occur again compared to a term about which we have no information. Second, the term frequency distributions are correlated. Indeed, no simple model appears capable of capturing the full meaning of text, hence our approach is a pragmatic one: to pick a model appropriate for the task at hand.

3.2 Rare events and laws of succession

The average English speaker uses about 20,000 of the 1,000,000 or more terms in an English dictionary [27]. In that sense, many terms that occur in documents are "rare events." This means that with reasonably small sample sets, we will see zero occurrences of many, many terms, and will still be required to estimate a non-zero value of $f(c,t)$. The maximum likelihood estimate, $f(c,t) = n(c,t)/n(c)$, is problematic: a class with $f(c,t) = 0$ will reject any document containing t.

Finding such estimates, also called *laws of succession*, has been pursued in classical statistics for centuries. Laplace showed that given the results of n tosses of a k-sided coin, i.e., the number of times each face occurred, $n_1, \ldots, n_k$, the correct Bayesian estimate for the probability of face i, denoted $\Pr_L(i|\{n_i\}, n)$, is not n_i/n, but $\frac{n_i+1}{n+k}$ [20]. This is the result of assuming that all possible associated k-component vectors of face probabilities $(p_1, \ldots, p_k)$ are *a priori* equally likely. This is called the *uniform prior* assumption. The above value of $\Pr_L(i|\{n_i\}, n)$ is obtained by using Bayes rule and evaluating $\frac{1}{\Pr[n_i]} \int_0^1 \theta \Pr[n_i|\theta] d\theta$. Alternative priors have been suggested and justified. We experimented with many of these, and found that Laplace's law wins by a few percent better classification accuracy all the time. For lack of space, we refer the reader to Ristad's paper for details [27]. With these adjustment, (and returning to our earlier notation) $f(c,t)$ is estimated as $(1+n(c,t))/(n(c)+L(c))$, where $L(c)$ is the size of the lexicon of class c.

3.3 Hierarchical classification

A *classifier* inputs a document and outputs a class. If the class is not the one from which the document was generated, we say the classifier *misclassified* that document. Typically, a classifier is *trained* by giving it example documents with class labels attached.

Our system has a classifier at each internal node in the taxonomy, with diverse feature sets. Given a new document d, the goal is to find a leaf node c such that the posterior $\Pr[c|d]$ is maximized among all leaves. The intuition behind doing this in levels rather than play off all the leaves is as follows. To give directions to a star in a distant galaxy, one can directly provide angular measures and a radial distance to that star. If these are noisy, one can end up far from the destination! If, on the other hand, one first somehow reaches the approximate center of the galaxy to which the star belongs (for which coarser navigation suffices) then the galaxy gets spread out around the traveler, and spotting the target star becomes easier.

The benefit may be lost if an error is made early in the process [18]. Thus a greedy search for the best leaf may be risky. Let the path to a leaf c from the root be $c_1, c_2, \ldots, c_k = c$. Since the root subsumes all classes, $\Pr[c_1|d] = 1$ for all d. Furthermore, we can write $\Pr[c_i|d] = \Pr[c_{i-1}|d] \Pr[c_i|c_{i-1}, d]$, for $i = 2, \ldots, k$. Taking logs, $\log \Pr[c_i|d] = \log \Pr[c_{i-1}|d] + \log \Pr[c_i|c_{i-1}, d]$. Suppose in the taxonomy we mark edge (c_{i-1}, c_i) with the edge cost $-\log \Pr[c_i|c_{i-1}, d]$. We are then seeking the least-cost path from the root c_1 to some leaf.

Computing the one-step conditional probability $\Pr[c_i|c_{i-1}, d]$ is straight-forward. For notational convenience, name c_{i-1} as r_0 and its children $\{r_j\}$. Then the probability that the document d belongs to the child node r_i given that it belongs to the parent node r_0 is given by: $\Pr[r_i|r_0, d] = \Pr[r_i|d] / \Pr[r_0|d]$ where $\Pr[r_0|d] = \sum_j \Pr[r_j|d]$ (where $\sum_j$ is over all the siblings of r_i. Note that $\Pr[r_i|d] = \Pr[d, r_i] / \sum_j \Pr[d, r_j]$

by Bayes rule. If we use the Bernoulli model as before, $\Pr[d|r_j] = \binom{n(d)}{\{n(d,t)\}} \prod_t f(r_j, t)^{n(d,t)}$. Care is needed here with finite-precision numbers, because the probabilities are very small (often less than 10^{-5000}) and the scaling needed to condition the probability prevents us from maintaining the numbers always in log-form.

3.4 Feature and noise terms

The above application of Bayes rule depended on a document model; this was embedded in the $f(c, t)$ parameters (and the independence assumption). We estimate these parameters during the training phase using sample documents. When building a model for each class c from a training set, we must decide if a term t appears only incidentally, or sufficiently consistently to suspect a causal connection; t is accordingly a *noise* term (also called a stopword) or a *feature* term. Given a new document, we should focus our attention only on the features for classifying it.

How can we pick the features from a hundred thousand terms in the lexicon? We are constrained both ways: we cannot miss the highly discriminating terms, and we cannot include everything, because the frequencies of some terms are noisy and unindicative of content. This is called the *feature-selection* problem in the statistical pattern recognition literature. Roughly speaking, we are in search of a set of terms that minimizes the probability that a document is misclassified, with the understanding that only terms in the intersection of the document and the feature set are used by the classifier.

It is not possible to search for the best feature set, because we don't know what the best possible classifier does, and because there are too many terms in the lexicon. So in practice we are interested in doing this for our fixed classifier. We want a heuristic that is essentially linear in the original number of terms, and makes preferably only one pass over the training corpus. We therefore restrict ourselves to the following approach: first we assign a merit measure to each term, then pick a prefix of terms with highest merit. This raises two questions: what measure, and what prefix? In answer to the first of these we use an index based *Fisher's linear discriminant.*

3.4.1 Fisher's discriminant

Suppose we are given two sets of points in n-dimensional Euclidean space, interpreted as two classes. Fisher's method finds a direction on which to project all the points so as to maximize (in the resulting one-dimensional space) the relative class separation as measured by the ratio of inter-class to intra-class variance. More specifically, let X and Y be the point sets, and μ_X, μ_Y be the respective centroids, i.e., $\mu_X = (\sum_X x)/|X|$ and $\mu_Y = (\sum_Y y)/|Y|$. Further, let the respective $n \times n$ covariance matrices be $\Sigma_X = (1/|X|)\sum_X (x-\mu_X)(x-\mu_X)^T$ and $\Sigma_Y = (1/|Y|)\sum_Y (y-\mu_Y)(y-\mu_Y)^T$.

Fisher's discriminant method seeks to find a vector α such that the ratio of the projected difference in means $|\alpha^T(\mu_X - \mu_Y)|$ to the average variance, $\frac{1}{2}\alpha^T(\Sigma_X + \Sigma_Y)\alpha = \alpha^T \Sigma \alpha$ is maximized. It can be shown that $\alpha = \Sigma^{-1}(\mu_X - \mu_Y)$ achieves the extremum when Σ^{-1} exists. Also, when X and Y are drawn from multivariate Gaussian distributions with $\Sigma_X = \Sigma_Y$, this is the optimal discriminator in that thresholding on $\alpha^T q$ for a test point q is the minimum error classifier [37].

Computing α involves a generalized eigenvalue problem involving the covariance matrices. In applications like signal processing where Fisher's discriminant is used, n is typically a few hundred at most; in the text domain, n is typically 50,000 to 100,000; and the covariance matrices may not be suitably sparse for efficient computation. Moreover, it is hard to interpret a discriminant that is a linear sum of term frequencies, possibly with negative coefficients! Our approach will be to take the directions α as given, namely, a coordinate axes for each term. We assign each term a figure of merit, which we call its *Fisher index*, based on the variance figures above, which is $\frac{|\alpha^T(\mu_X - \mu_Y)|}{\alpha^T \Sigma \alpha}$ in the two-class case. For each term t, $\alpha = e_t$ is a unit vector in the direction of t.

In general, given a set of two or more classes $\{c\}$, with $|c|$ documents in class c, we compute the ratio of the so-called between-class to within-class scatter. Switching back to our term frequency notations, we express this as:

$$\mathsf{Fisher}(t) = \frac{\sum_{c_1,c_2}\Big(\mu(c_1,t) - \mu(c_2,t)\Big)^2}{\sum_c \frac{1}{|c|}\Big(x(d,t) - \mu(c,t)\Big)^2}, \quad (2)$$

$$\text{where} \quad \mu(c,t) = \tfrac{1}{|c|}\textstyle\sum_{d\in c} x(d,t). \quad (3)$$

The information theory literature provides some other notions of good discriminants. One of the best known is *mutual information* [6]. Closer inspection shows that its computation is more complicated and not as easily amenable to the optimizations described next.

3.4.2 Selecting a cut-off

Let F be the list of terms in our lexicon sorted by decreasing Fisher index. Our heuristic is to pick from F a prefix F_k of the k most discriminating terms. F_k must include most features and exclude most noise terms. A short F_k enables fast classification and holding a larger taxonomy in memory. Too large an F_k will fit the training data very well, but will result in degraded accuracy for test data, due to *overfitting*. There are various techniques for pruning feature sets. We use *validation*, some others approaches are to use minimum description length principle, resampling or cross validation. We randomly partition the pre-classified samples into $\mathcal{T}$, the *training set* and $\mathcal{V}$, the *validation set*. We compute the Fisher index of each term based on $\mathcal{T}$, and then classify $\mathcal{V}$ using various prefixes F_k. Let N_k be the number of misclassified documents using F_k; then we seek k for which N_k is minimized.

For classification we choose the class c that maximizes the following *a priori* class probability based on the Bernoulli model introduced in Section 3.1:

$$\Pr[c|d, F_k] = \frac{\pi(c)\prod_{t\in d\cap F_k} f(c,t)^{n(d,t)}}{\sum_{c'} \pi(c')\prod_{t\in d\cap F_k} f(c',t)^{n(d,t)}}, \quad (4)$$

where π is the prior distribution on the classes. This is a special case of the "naive" Bayes rule[9] Because we obtain a steep reduction in retained features it is possible to hold in memory the $f(c,t)$ tables needed for classification, even for a very large taxonomy such as our Yahoo! sample with over 370 nodes.

3.4.3 Pseudocode

Although the computation of Fisher indices (2) appears simple, care is needed for memory management when the corpus has hundreds of nodes in the taxonomy and a large lexicon. For each term t and class c, we need to maintain a map H to $\sum_{d\in c} x(d,t)$ and $\sum_{d\in c} x(d,t)^2$. With 100,000 terms and 300 classes, we must exploit sparseness; so we hash on t and point to a sparse vector indexed by c. In spite of sparseness, H ranges into hundreds of megabytes, and therefore updates must be staged out, which we do in the usual way:

```
Initialize in-memory hash table H_M and disk table H_D
Loop over documents in the corpus:
    If H_M is larger than available memory
        Make a sequential pass over H_D:
            Update H_D(t) with values in H_M(t)
        Clear H_M
    Update H_M with the next document
Sort H_D by Fisher index value to obtain F, the feature list.
```

Note how the additive nature of the statistics we maintain makes this simple. This also makes it easy to insert, delete, and move documents dynamically from the collection.

Another essential aspect of the implementation is that it computes the best cut-off in only one pass over the document samples. Fix a test document $d \in \mathcal{V}$ and consider what happens as we grow the prefix k. Typically, d will be misclassified upto some prefix because there aren't enough discriminating terms in the prefix, and then at some point to get correctly classified. For some documents, at a larger prefix, a noise term gets into the feature set and misclassifies the document again. Let this 0-1 function (1 iff d is misclassified) be $N_k(d)$; then we seek to find $\sum_d N_k(d)$ for all k. We will hold N_k in memory, and make a single pass through $\mathcal{V}$.

```
Initialize N_k = 0 for k = 0, 1, ...
Compute f(c,t), μ(c,t), and feature list F as before
For each document d ∈ V
    Let c', c'' be children of c
    Compute Pr[c'|c,d,F_0] = π(c')/Σ_{c''} π(c'') for all c'
    For k = 1, 2, ...
        Find Pr[c'|c,d,F_k] using Pr[c'|c,d,F_{k-1}] and (4)
        Suppose the true class for d is child c_* of c.
        If Pr[c_*|c,d,F_k] < max_{c'} Pr[c'|c,d,F_k]
            N_k ← N_k + 1
```

Note that to be exact, we have to compute $N_k(d)$ for all k, not only for those k for which $t_k \in d$, since given F_k, $f(c,t)$ depends on $n(c,t)$ and $\sum_{t'\in F_k} n(c,t')$.

3.5 Extracting document signatures

Up to a point, the user can sift a query response based only on the topic paths. However, even the leaf classes are necessarily coarser than individual documents; support is therefore needed to browse quickly through many documents without looking into the documents in detail. Most search engines attach a few lines from each document. Often these are the title and first few lines; or they are sentences with the most search terms. For many documents, better keyword extraction is needed. Moreover, as we have argued, these signatures should be extracted relative to a node in the taxonomy.

Given this reference node c, one approach is to concatenate the training documents associated with c into a super document d, and then rank terms $t \in d$ in decreasing order of the number of standard deviations that $x(d,t)$ is away from $f(c,t)$. Here our earlier simplistic document model gets into trouble: as mentioned on page 4, a term that has occurred once in a document is more likely to occur again. Since the Bernoulli model does not take this into account, frequent terms often remain surprising all along the taxonomy path.

Matters are improved by moving to another simple model. First suppose we have a single test document d, and consider $t \in d$. If the observed fraction of training documents in class c containing term t is $\theta(c,t)$, we simply sort all $t \in d$ by increasing $\theta(c,t)$ and report the top few. If there are $\ell > 1$ test documents in c, we find the fraction $\phi(t)$ that contains t, and sort the t's in increasing order of $\frac{(\theta(c,t)-\phi(t))\sqrt{\ell}}{\sqrt{\theta(c,t)(1-\theta(c,t))}}$. Both in fact correspond to P-values computed using the normal approximation to the binomial distribution.

[9] "Naive" in that we again neglect dependences between term frequencies.

Patent:	Communication:	329 Modulator
		332 Demodulator
		343 Antenna
		379 Telephony
	Electricity:	307 Transmission
		318 Motive
		323 Regulator
		219 Heating
	Electronics:	330 Amplifier
		331 Oscillator
		338 Resistor
		361 System

Figure 1: The portion of the US Patent database taxonomy that we used for our experiments, with numeric class codes.

4 Performance evaluation

In this section we study the performance of our system. There are three aspects to performance: first, to what extent this paradigm assists in text search and browsing; second, how accurate our techniques for feature selection and hierarchical classification are; and third, how efficient or scalable our system is. The first item is at this point a matter of qualitative judgement, as is the evaluation of the signature-finding techniques. The quality of feature selection and classification can be measured precisely, and we present these results here. As regards efficiency, we restrict our discussion to quoting our running times on a specific platform, and show that they scale favorably with corpus size.

4.1 Datasets and measures

We used three data sources: the Reuters benchmark used widely in the IR community, the US Patent database, hereafter referred to as USPatent, and Yahoo!. For evaluation, the simple scenario is a m-class problem where each document belongs to exactly one class. We can draw up a $m \times m$ contingency table, entry (i, j) showing how many test documents of class i were judged to be of class j. This is called the *confusion matrix*. One important number to compute from a confusion matrix is the sum of diagonal entries divided by the sum of all elements: this gives the fraction of documents correctly classified. If each document has exactly one class, this number is the same as *microaveraged recall* and *precision* as defined by Lewis [21]. Matters are complicated by documents having multiple classes. Due to space constraint we omit our experiments with this setting. See Lewis for more details on evaluating classifiers [21].

4.2 Evaluation of feature selection

Although Reuters has provided a taxonomy for its articles, the data available does not include taxonomy codes in the class header. For this subsection we will work with other corpora where such information is explicitly provided.

The sample of USPatent that we used has three nodes in the first level, Communication, Electricity and Electronics. Each has four children in the second level. Figure 1 shows the taxonomy we used. The overlap in vocabulary between some of the nodes, e.g., modulator, demodulator, amplifier, oscillator; and motive, heating, resistor make the classification task appear more challenging than Reuters, which deals with a more diverse set of topics.

Figure 2 shows the results of validation experiments over the patent database. 500 training patents and 300 validation patents were picked at random from each of the 12 leaves in Figure 1. The Fisher index ordering gives rapid reduction in classification error within just a few hundred feature terms, out of the roughly 30,000 terms in our lexicon. For some classes the error goes up slightly (not visible in the range shown) after a minimum due to overfitting. The smallest minima and corresponding errors are roughly at 160 terms, 25.1% for Patent; 200 terms, 11.7% for Communication; 890 terms, 17.8% for Electricity; and 9130 terms, 16.6% for Electronics. The minima are not very sharp, but the diversity of the feature set sizes still questions the common practice of picking a fixed number of most frequent terms in each class as features. We list the best features in the patent taxonomy below; notice how the sets change down the levels.

Patent: Signal, modulate, motor, receive, antenna, telephone, transmit, frequency, modulation, modulator, demodulator, current, voltage, data, carrier, power, amplifier, phase, call, amplitude.

Patent:Communication: Antenna, telephone, modulator, demodulator, signal, modulate, output, call, modulation, input, demodulated, frequency, phase, communication, radar, demodulating, space, detector, line, demodulation, transmit, circuit.

Patent:Electricity: Motor, heat, voltage, transistor, output, circuit, connect, input, weld, extend, surface, current, position, gate, speed, control, terminal, drive, regulator, signal, rotor.

Patent:Electronics: Amplifier, oscillator, input, output, frequency, transistor, signal, laser, emitter, couple, amplify, gain, resistance, connect, extend, form, contact, differential, material, resistor.

4.3 The Reuters benchmark

The Reuters benchmark has about 7,700 training documents and 3,500 testing documents from about 135 classes. Each document is about 200 terms long on average. We experimented with Reuters to ensure that our basic classifier is of acceptable quality. Less than a tenth of the articles are assigned multiple classes. In fact, in some cases, some class labels were refinements of others, e.g., *grain* and *wheat*, and it would be incorrect to regard them as classes at the same level since some classes imply others. For simplicity, we just removed all but the first class label from each article. Alternatively, for m classes, one can build m two-way classifiers; the c-th classifier discriminating between c and $\bar{c}$.

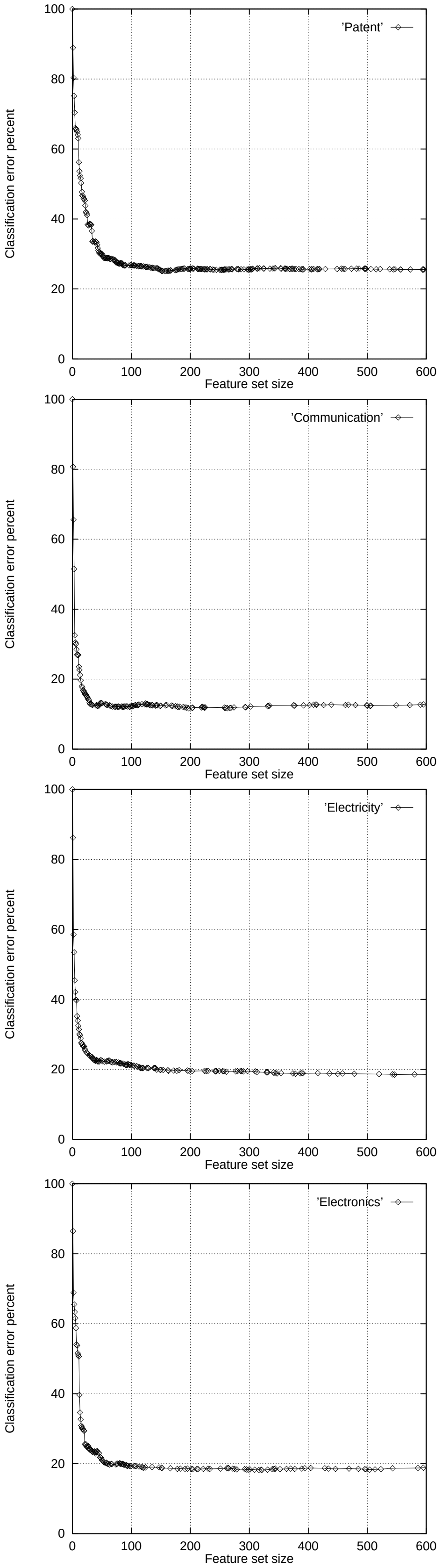

Figure 2: Evaluation of feature selection.

Fraction	.01	.05	.10	.20	1.0
Accuracy	.43	.56	.59	.62	.66

Table 1: Verification that the training set is adequate. The first row gives the fraction of the training set sampled for training. The second row gives micro-averaged recall.

We only considered classes with at least 20 training documents. Only 30 classes were large enough, giving a 30×30 confusion matrix. The best accuracy was achieved using about 8000 features. The best microaveraged recall/precision is 0.87, which compares favorably with previous experiments [2], although those studies used the $(c, \bar{c})$ style classifiers. The numbers are not therefore directly comparable, although since very few documents had multiple topics, we expect similar results had the earlier experiments used only the single-topic documents.

For this benchmark there is no benefit from hierarchy. To test the effect of our feature selection, we compared it with an implementation that performs singular value decomposition (SVD) on the original term-document matrix, projects documents down to a lower dimensional space, and uses a Bayesian classifier in that space assuming the Gaussian distribution [32]. Our classifier was more accurate by 10–15%, in spite of its simplicity. Our explanation is that the SVD, in ignoring the class labels, finds projections along directions of large variance, which may not coincide with directions of best separation between documents with different classes.

4.4 Evaluation of hierarchical classification

In this section we focus on the hierarchical USPatent dataset. First we ensure that the training set is adequate. To do this, we train using different sized random samples from the training set and verify that the test performance converges. This is shown in Table 1. Next we compare the hierarchical classifier with a standard vector-space [29] based classifier. Each document is a vector in term space; each class is the sum or centroid of its document vectors. The similarity between two vectors is their cosine. Weighting the terms usually results in better relevance ranking. There are over 287 variants of term weighting schemes with tuned magic constants reported [29]. We pick one version recommended by Sparck-Jones [13, 16].

$$
\begin{aligned}
n_{\max}(d) &= \max_{t \in d} n(d,t) \\
m &= \text{number of classes} \\
n_t &= \textstyle\sum_c \operatorname{sign}(n(c,t)) \\
w(c,t) &= \left(1 + \tfrac{n(d,t)}{n_{\max}(d)}\right)\left(1 + \lg \tfrac{m}{n_t}\right) \\
\mathsf{Score}(c,d) &= \frac{\vec{x}_d \cdot \vec{w}_c}{|\vec{x}_d|\;|\vec{w}_c|}
\end{aligned}
$$

Class name	329	332	343	379	307	318	323	219	330	331	338	361
329_Demodulator	51	9	2	1	0	6	5	2	12	11	0	2
332_Modulator	21	27	3	2	3	7	6	4	10	12	2	2
343_Antennas	10	6	47	3	4	2	6	1	1	4	14	3
379_Telephony	9	1	1	65	1	5	2	5	3	1	5	3
307_Transmission	1	1	1	1	57	2	8	5	0	1	19	4
318_Motive	6	4	1	1	1	41	7	13	14	4	2	3
323_Regulator	8	3	1	3	7	4	59	7	2	1	2	5
219_Heating	2	1	0	0	0	18	9	49	12	1	2	5
330_Amplifier	6	5	1	0	1	17	1	8	53	3	4	1
331_Oscillator	10	2	3	0	6	9	4	7	10	33	13	4
338_Resistor	0	0	0	0	3	0	3	2	0	0	87	4
361_System	2	1	1	1	9	8	8	9	1	1	30	29

Table 2: Confusion matrix for the weighted cosine one-level classifier. Each row sums to 100, modulo rounding. The diagonal elements add up to only 0.48 of the total number of the documents. This is the micro-averaged recall.

Class name	329	332	343	379	307	318	323	219	330	331	338	361
329_Demodulator	80	5	0	0	0	2	0	3	5	4	0	0
332_Modulator	16	55	1	0	1	2	1	3	9	11	0	0
343_Antennas	5	5	63	1	1	0	2	0	0	2	15	6
379_Telephony	4	2	1	82	0	1	0	2	1	1	1	4
307_Transmission	0	0	0	0	55	2	3	3	0	2	26	8
318_Motive	6	4	0	2	3	48	5	16	8	5	1	2
323_Regulator	3	1	1	2	3	2	81	6	0	0	1	1
219_Heating	1	1	0	0	0	10	4	72	7	0	3	1
330_Amplifier	3	9	0	0	0	10	0	11	57	8	0	1
331_Oscillator	15	8	0	0	0	4	0	7	8	47	5	4
338_Resistor	0	0	0	0	1	0	2	0	1	0	92	4
361_System	1	0	0	0	2	6	6	10	1	1	12	61

Table 3: Confusion matrix for our multi-level classifier, showing much larger diagonal elements, i.e, more frequently correct classification. The micro-averaged recall is 0.66.

We see a substantial gain in accuracy over the standard weighted-cosine classifier. We did further experiments to see how much of the gains was from feature selection as against the hierarchy. To do this, we can fix the feature selection and classification modules, and only change the taxonomy: one will be the taxonomy in Figure 1, the other will have the root and the 12 leaves. We have to be very careful to make this a fair competition, making sure that the class models are represented with the same complexity (number of parameters) in the two settings. In counting the number of parameters we must also account for the sparsity of the term frequency tables; we have no direct control on this. By trial and error, we came up with the comparative evaluation shown in Table 4.

In this dataset, the accuracy benefit from hierarchy is modest compared to the benefit from feature selection. However, note that the flat classifier has a steep performance penalty because it has to compare too many classes all at once. How to allocate a fixed number of parameters among the taxonomy nodes for best overall classification is an interesting issue.

Summarizing, we showed that our feature selection is effective, and that our classifier is significantly more accurate than cosine-based ones and comparable to the best known for flat sets of classes. Hierarchy enhances accuracy in modest amounts, but greatly increases speed.

Classifier	Prefix	Parameters	Recall	Time/doc
Flat	250	2651	0.60	15 ms
Taxonomy	950,200,400,800	2649	0.63	6 ms

Table 4: The benefits of hierarchy. The prefix field in the second row correspond to the four internal nodes in the USPatent tree.

5 Related work

We survey the following overlapping areas of related research: IR systems and text databases, data mining, statistical pattern recognition, and machine learning.

IR systems and text databases. The most mature ideas in IR, which are also those successfully integrated into commercial text search systems such as Verity[10], ConText[11] and AltaVista, involve processing at a relatively syntactic level; e.g., stopword filtering, tokenizing, stemming, building inverted indices, computing heuristic term weights, and computing similarity measures between documents and queries in the vector-space model [33, 30, 11]. Singular value decomposition on the term-document matrix has been found to cluster semantically related documents together even if they do not share keywords [8, 24].

We deviate from these systems in the paradigm of interactive exploration and dynamic reorganization of query response based on user context. A closely related work with respect to the user interface is Scatter-Gather [7], which alternates between the system presenting a document cluster and a user selecting a subset from them. Topic paths in the taxonomy have some advantages: they can be precomputed and queried upon, and the path annotations (as in Yahoo!) are often more informative than "cluster digests" generated automatically.

Data mining, machine learning, and pattern recognition. More recent work on text includes statistical modeling of documents, clustering and classification, thesaurus generation by term associations, and query expansion [31, 10, 34]. The supervised classification problem has been addressed in statistical decision theory (both classical [35] and Bayesian[3]), statistical pattern recognition [9, 12] and machine learning [36, 23, 19]. Classifier can be parametric or nonparametric. Two well-known classes of non-parametric classifiers are decision trees, such as CART [4] and C4.5 [25], and neural networks [14, 22, 15]. For such classifiers, feature sets larger than 100 are considered extremely large. Document classification may require more than 50,000.

Impressive accuracy has been obtained using decision rules induction on the Reuters dataset [2]. They

[10] `http://www.verity.com`
[11] `http://www.oracle.com/products/oracle7/oracle7.3/html/context_qa.html`

learn boolean formulas involving the presence or absence of terms in documents, after pruning a static stopword set and picking the most frequent terms from each class as features. The learning algorithm does cut down this set further, but needs computation to do so. More advanced heuristics have been proposed [17]. For virtually all of these approaches, the complexity in the number of features is supralinear (e.g., quadratic in the number of starting features and exponential in the degree of dependence between features), which makes their use difficult for large numbers of features and gigabyte-sized corpora. As a result, these advanced methods have been typically restricted to under a thousand features and documents [18]; however, that work does add evidence that exploiting taxonomies is a valuable strategy.

6 Conclusion

We have demonstrated that hierarchical views of text databases can improve search and navigation in many ways, and presented some of the tools needed to maintain and navigate in such a database. A combination of hierarchy, feature selection, and context-sensitive document signatures greatly enhanced the searching experience. In future we plan to experiment more extensively with larger taxonomies and larger corpora for better confidence in the statistical procedures. It is also of interest to develop classifiers that maximize accuracy given bounded space, i.e., model size, and bounded time. Table 4 suggests the interesting problem of allocating the total space among nodes of a hierarchy for best overall accuracy. Finding means to measure the effectiveness of the signature finding schemes also appears worthwhile.

References

[1] P. Anick and S. Vaithyanathan. Exploiting clustering and phrases for context-based information retrieval. In *SIGIR*, 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] J. O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer-Verlag, New York; Berlin, 1985. ISBN: 0-387-96098-8.

[4] 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.

[5] C. Chekuri, M. Goldwasser, P. Raghavan, and E. Upfal. Web search using automatic classification. Submitted for publication, 1996.

[6] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley and Sons, Inc., 1991.

[7] D. R. Cutting, D. R. Karger, and J. O. Pedersen. Constant interaction-time scatter/gather browsing of very large document collections. In *SIGIR*, 1993.

[8] S. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the Society for Information Science*, 41(6):391–407, 1990.

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

[10] C. Falaoutsos and D. W. Oard. A survey of information retrieval and filtering methods. Technical Report CS-TR-3514, University of Maryland, College Park, MD 20742, Aug. 1995.

[11] W. B. Frakes and R. Baeza-Yates. *Information retrieval: Data structures and algorithms*. Prentice-Hall, 1992.

[12] K. Fukunaga. *An Introduction to Statistical Pattern Recognition, 2nd ed.* Academic Press, New York, 1990.

[13] D. Harman. Ranking algorithms. In W. B. Frakes and R. Baeza-Yates, editors, *Information retrieval: Data structures and algorithms*, chapter 14. Prentice-Hall, 1992.

[14] D. R. Hush and B. G. Horne. Progress in supervised neural networks. *IEEE Signal Processing Magazine*, pages 8–39, January 1993.

[15] A. K. Jain, J. Mao, and K. Mohiuddin. Artificial neural networks: A tutorial. *Computer*, 29(3):31–44, March 1996.

[16] K. S. Jones. A statistical interpretation of term specificity and its applications in retrieval. *Journal of Documentation*, 28(1):11–20, 1972.

[17] D. Koller and M. Sahami. Toward optimal feature selection. In L. Saitta, editor, *International Conference on Machine Learning*, volume 13. Morgan-Kaufmann, 1996.

[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] P. Langley. *Elements of Machine Learning*. Morgan Kaufmann, 1996. ISBN: 1-55860-301-8.

[20] P.-S. Laplace. *Philosophical Essays on Probabilities*. Springer-Verlag, New York, 1995. Translated by A. I. Dale from the 5th French edition of 1825.

[21] D. Lewis. Evaluating text categorization. In *Proceedings of the Speech and Natural Language Workshop*, pages 312–318. Morgan-Kaufmann, Feb. 1991.

[22] R. P. Lippmann. Pattern classification using neural networks. *IEEE Communications Magazine*, pages 47–64, Nov. 1989.

[23] B. K. Natarajan. *Machine Learning: A Theoretical Approach*. Morgan-Kaufmann, 1991.

[24] C. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala. Latent sematic indexing: A probabilistic analysis. Submitted for publication.

[25] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[26] P. Raghavan. Information retrieval algorithms: A survey. In *Symposium on Discrete Algorithms*. ACM-SIAM, 1997. Invited paper.

[27] E. S. Ristad. A natural law of succession. Research report CS-TR-495-95, Princeton University, July 1995.

[28] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In *SIGIR*, pages 232–241, 1994.

[29] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

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

[31] H. Schutze, D. A. Hull, and J. O. Pederson. A comparison of classifiers and document representations for the routing problem. In *SIGIR*, pages 229–237, 1995.

[32] S. Vaithyanathan. Document classification using principal component analysis. Personal communication, May 1997.

[33] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 1979. Also available on-line at http://www.dcs.gla.ac.uk/Keith/Preface.html.

[34] E. M. Voorhees. Using WordNet to disambiguate word senses for text retrieval. In *SIGIR*, pages 171–180, 1993.

[35] A. Wald. *Statistical Decision Functions*. Wiley, New York, 1950.

[36] S. M. Weiss and C. A. Kulikowski. *Computer Systems That Learn*. Morgan-Kaufmann, 1990.

[37] T. Y. Young and T. W. Calvert. *Classification, Estimation and Pattern Recognition*. Elsevier, 1974.

[38] G. K. Zipf. *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*. Addison-Wesley, 1949.