Data mining for hypertext: A tutorial survey

Soumen Chakrabarti*
Indian Institute of Technology Bombay†
soumen@cse.iitb.ernet.in

ABSTRACT

With over 800 million pages covering most areas of human endeavor, the World-wide Web is a fertile ground for data mining research to make a difference to the effectiveness of information search. Today, Web surfers access the Web through two dominant interfaces: clicking on hyperlinks and searching via keyword queries. This process is often tentative and unsatisfactory. Better support is needed for expressing one's information need and dealing with a search result in more structured ways than available now. Data mining and machine learning have significant roles to play towards this end.

In this paper we will survey recent advances in learning and mining problems related to hypertext in general and the Web in particular. We will review the continuum of supervised to semi-supervised to unsupervised learning problems, highlight the specific challenges which distinguish data mining in the hypertext domain from data mining in the context of data warehouses, and summarize the key areas of recent and ongoing research.

1 Introduction

The volume of unstructured text and hypertext data exceeds that of structured data. Text and hypertext are used for digital libraries, product catalogs, reviews, newsgroups, medical reports, customer service reports, and homepages for individuals, organizations, and projects. Hypertext has been widely used long before the popularization of the Web. Communities such as the **ACM**[1] Special Interest Groups on **Information Retrieval (SIGIR)**[2], **Hypertext, Hypermedia and Web (SIGLINK/SIGWEB)**[3] and **Digital Libraries**[4] have been engaged in research on effective search and retrieval from text and hypertext databases.

The spectacular ascent in the size and popularity of the Web has subjected traditional information retrieval (IR) techniques to an intense stress-test. The Web exceeds 800 million HTML pages, or six terabytes of data on about three million servers. Almost a million pages are added daily, a typical page changes in a few months, and several hundred gigabytes change every month. Even the largest search engines, such as **Alta Vista**[5] and **HotBot**[6] index less than 18% of the accessible Web as of February 1999 [49], down from 35% in late 1997 [6].

Apart from sheer size and flux, the Web and its users differ significantly from traditional IR corpora and workloads. The Web is populist hypermedia at its best: there is no consistent standard or style of authorship dictated centrally; content is created autonomously by diverse people. Misrepresentation of content by 'spamming' the page with meaningless terms is rampant so that keyword indexing search engines rank pages highly for many queries. Hyperlinks are created for navigation, endorsement, citation, criticism, or plain whim.

Search technology inherited from the world of IR is evolving slowly to meet these new challenges. As mentioned before, search engines no longer attempt to index the whole Web. But apart from this deficiency of scale, search engines are also known for poor accuracy: they have both low *recall* (fraction of relevant documents that are retrieved) and low *precision* (fraction of retrieved documents that are relevant). The usual problems of text search, such as synonymy, polysemy (a word with more than one meaning), and context sensitivity become especially severe on the Web. Moreover, a new dimension is added because of extant partial structure: whereas IR research deals predominantly with *documents*, the Web offers *semistructured* data in the form of directed graphs where nodes are primitive or compound objects and edges are field labels.

Apart from the IR community, documents and the use of natural language (NL) have been studied also by a large body of linguists and computational linguists. NL techniques can now parse relatively well-formed sentences in many languages [36, 71, 28, 68], disambiguate polysemous words with high accuracy [3, 62, 11], tag words in running text with part-of-speech information [3, 32], represent NL documents in a canonical machine-usable form [67, 40, 65], and perform NL translation [39, 5, 4]. A combination of NL and IR techniques have been used for creating hyperlinks automatically [9, 2, 35, 46, 12] and expanding brief queries with related words for enhanced search. For reasons unclear to us, popular Web search engines have been slow to embrace these capabilities. Although we see the progression towards better semantic understanding as inevitable, NL techniques are outside the scope of this survey.

[1] http://www.acm.org
[2] http://www.acm.org/sigir
[3] http://www.acm.org/siglink
[4] http://www.acm.org/dl
[5] http://www.altavista.com
[6] http://www.hotbot.com

Outline: In this survey we will concentrate on statistical techniques for learning structure in various forms from text, hypertext and semistructured data.

Models: In §2 we describe some of the models used to represent hypertext and semistructured data.

Supervised learning: In §3 we discuss techniques for supervised learning or *classification*.

Unsupervised learning: The other end of the spectrum, unsupervised learning or *clustering* is discussed in §4.

Semi-supervised learning: Real-life applications fit somewhere in between fully supervised or unsupervised learning, and are discussed in §5.

Social network analysis: A key feature which distinguishes hypertext mining from warehouse mining and much of IR is the analysis of structural information in hyperlinks. This is discussed in §6.

2 Basic models

In its full generality, a model for text must build machine representations of world knowledge, and must therefore involve a NL grammar. Since we restrict our scope to statistical analyses, we need to find suitable representations for text, hypertext, and semistructured data which will suffice for our learning applications. We discuss some such models in this section.

2.1 Models for text

In the IR domain, documents have been traditionally represented in the **vector space model** [63, 29]. Documents are tokenized using simple syntactic rules (such as whitespace delimiters in English) and tokens *stemmed* to canonical form (e.g., 'reading' to 'read,' 'is,' 'was,' 'are' to 'be'). Each canonical token represents an axis in a Euclidean space. Documents are vectors in this space. In the crude form, if a term t occurs $n(d,t)$ times in document d, the t-th coordinate of d is simply $n(d,t)$. One may choose to normalize the length of the document to 1, typically in the L_1, L_2, or L_∞ norms: $||d||_1 = \sum_t n(d,t)$, $||d||_2 = \sqrt{\sum_t n(d,t)^2}$, and $||d||_\infty = \max_t n(d,t)$.

These representations do not capture the fact that some terms (like 'algebra') are more important than others (like 'the' and 'is') in determining document content. If t occurs in N_t out of N documents, N_t/N gives a sense of the rarity, hence, importance, of the term. The "inverse document frequency" $\mathrm{IDF}(t) = 1 + \log \frac{N_t}{N}$ is used to stretch the axes of the vector space differentially. (Many variants of this formula exist; our form is merely illustrative.) Thus the t-th coordinate of d may be chosen as $\frac{n(d,t)}{||d||_1} \times \mathrm{IDF}(t)$, popularly called the 'TFIDF' (term frequency times inverse document frequency) weighted vector space model.

Alternatively, one may construct probabilstic models for document generation. Once again, we should start with the disclaimer that these models have no bearing on grammar and semantic coherence. The simplest statistical model is the **binary** model. In this model, a document is a set of terms, which is a subset of the *lexicon* (the universe of possible terms). Word counts are not significant. In the **multinomial** model, one imagines a die with as many faces as there are words in the lexicon, each face t having an associated probability θ_t of showing up when tossed. To compose a document the author fixes a total word count (arbitrarily or by generating it from a distribution), and then tosses the die as many times, each time writing down the term corresponding to the face that shows up.

In spite of being extremely crude and not capturing any aspect of language or semantics, these models often perform well for their intended purpose. In spite of minor variations all these models regard documents as multisets of terms, without paying attention to ordering between terms. Therefore they are collectively called **bag-of-words** models.

2.2 Models for hypertext

Hypertext has hyperlinks in addition to text. These are modeled with varying levels of detail, depending on the application. In the simplest model, hypertext can be regarded as a directed graph (D, L) where D is the set of nodes, documents, or pages, and L is the set of links. Crude models may not need to include the text models at the nodes. More refined models will characterize some sort of joint distribution between the term distribution of a node with those in a certain neighborhood. One may also wish to recognize that the source document is in fact a *sequence* of terms interspersed with outbound hyperlinks. This may be used to establish specific relations between certain links and terms (§6).

On occasion we will regard documents as being generated from topic-specific term distributions. For example, the term distribution of documents related to 'bicycling' is quite different from that of documents related to 'archaeology.' Unlike journals on archaeology and magazines on bicycling, the Web is not isolated: nodes related to diverse topics link to each other. (We have found recreational bicycling pages to link to pages about first-aid significantly more often than a typical page.) If the application so warrants (as in §6.2) one needs to model such coupling among topics.

2.3 Models for semistructured data

Apart from hyperlinks, other structures exist on the Web, both across and within documents. One prominent kind of inter-document structure are topic directories like the **Open Directory Project**[7] and **Yahoo!**[8]. Such services have constructed, through human effort, a giant taxonomy of topic directories. Each directory has a collection of hyperlinks to relevant (and often popular or authoritative) sites related to the specific topic. One may model tree-structured hierarchies with an `is-a(specific-topic, general-topic)` relation, and an `example(topic, url)` relation to assign URLs to topics. Although topic taxonomies are a special case of semistructured data, it is important and frequent enough to mention separately.

Semistructured data is a point of convergence [25] for the **Web**[9] and **database**[10] communities: the former deals with *documents*, the latter with *data*. The form of that data is evolving from rigidly structured relational tables with numbers and strings to enable the natural representation of complex real-world objects like books, papers, movies, jet engine

[7] `http://dmoz.org`
[8] `http://www.yahoo.com`
[9] `http://www9.org`
[10] `http://www.acm.org/sigmod`

components, and chip designs without sending the application writer into contortions. Emergent representations for semistructured data (such as **XML**[11]) are variations on the **Object Exchange Model (OEM)**[12] [54, 33]. In OEM, data is in the form of atomic or compound *objects*: atomic objects may be integers or strings; compound objects refer to other objects through labeled edges. HTML is a special case of such 'intra-document' structure.

The above forms of irregular structures naturally encourage data mining techniques from the domain of 'standard' structured warehouses to be applied, adapted, and extended to discover useful patterns from semistructured sources as well.

3 Supervised learning

In supervised learning, also called classification, the learner first receives training data in which each item is marked with a label or class from a discrete finite set. The algorithm is trained using this data, after which it is given unlabeled data and has to guess the label.

Classification has numerous applications in the hypertext and semistructured data domains. Surfers use topic directories because they help structure and restrict keyword searches, making it easy, for example, to ask for documents with the word `socks` which are about garments, not the **security proxy protocol**[13]. (`socks AND NOT network` will drop garment sites boasting of a large distribution network.) Robust hypertext classification is useful for email and newsgroup management and maintining Web directories.

As another example, a campus search engine that can categorize faculty, student and project pages can answer queries that select on these attributes in addition to keywords (find faculty interested in cycling). Furthermore, learning relations between these types of pages (advised-by, investigator-of) can enable powerful searches such as "find faculty supervising more than the departmental average of Masters students."

3.1 Probabilistic models for text learning

Suppose there are m classes or topics $c_1, \ldots, c_m$, with some training documents D_c associated with each class c. The prior probability of a class is usually estimated as the fraction of documents in it, i.e., $|D_c|/\sum_c |D_c|$. Let T be the universe of terms, or the *vocabulary*, in all the training documents.

3.1.1 Naive Bayes classification

The bag-of-words models readily enable naive Bayes classification [13] ('naive' indicates the assumption of independence between terms).

One may assume that for each class c, there is a binary text generator model. The model parameters are $\phi_{c,t}$ which indicates the probability that a document in class c will mention term t at least once. With this definition,

$$\Pr(d|c) \quad = \quad \prod_{t \in d} \phi_{c,t} \prod_{t \in T, t \notin d} (1 - \phi_{c,t}). \qquad (1)$$

Since typically $|T| \gg |d|$, short documents are discouraged by this model. Also, the second product makes strong independence assumptions and is likely to greatly distort the estimate of $\Pr(d|c)$.

In the multinomial model, each class has an associated die with T faces. The ϕ parameters are replaced with $\theta_{c,t}$, the probability of the face $t \in T$ turning up on tossing the die (§2.1). The conditional probability of document d being generated from a class c is

$$\Pr(d|c) \quad = \quad \binom{||d||_1}{\{n(d,t)\}} \prod_{t \in d} \theta_{c,t}^{n(d,t)}, \qquad (2)$$

where $\binom{||d||_1}{\{n(d,t)\}} = \frac{||d||_1!}{n(d,t_1)!\, n(d,t_2)! \cdots}$ is the multinomial coefficient. Note that short documents are encouraged, in particular, the empty document is most likely. Not only is inter-term correlation ignored, but each occurrence of term t results in a multiplicative $\theta_{c,t}$ 'surprise' factor. The multinomial model is usually prefered to the binary model because the additional term count information usually leads to higher accuracy [51, 14].

Both models are surprisingly effective given their crudeness; although their estimate of $\Pr(d|c)$ must be terrible, their real task is to pick $\arg\max_c \Pr(c|d)$, and the poor estimate of $\Pr(d|c)$ appears to matter less than one would expect [31, 26].

3.1.2 Parameter smoothing and feature selection

Many of the terms $t \in T$ will occur in only a few classes. The maximum likelihood (ML) estimates of $\theta_{c,t}$ (defined as $\sum_{d \in D_c} n(d,t) / \sum_\tau \sum_{d \in D_c} n(d,\tau)$) will be zero for all classes c where t does not occur, and so $\Pr(c|d)$ will also be zero even if a test document d contains even one such term. To avoid this in practice, the ML estimate is often replaced by the Laplace corrected estimate [47, 61]:

$$\theta_{c,t} \quad = \quad \frac{1 + \sum_{d \in D_c} n(d,t)}{|T| + \sum_\tau \sum_{d \in D_c} n(d,\tau)}. \qquad (3)$$

Since in the binary case there are two outcomes instead of T outcomes, the corresponding correction is

$$\phi_{c,t} \quad = \quad \frac{1 + |\{d \in D_c : t \in d\}|}{2 + |D_c|}. \qquad (4)$$

Unfortunately the second form often leads to a gross overestimation of $\Pr(t|c)$ for sparse classes (small D_c). Other techniques for parameter smoothing, which use information from related classes (e.g., parent or sibling in a topic hierarchy) are discussed in §*3.1.5*.

Many terms in T are quite useless and even potentially harmful for classification. 'Stopwords' such as 'a,' 'an,' 'the,' are likely candidates, although one should exercise caution in such judgement: 'can' (verb) is often regarded as a stopword, but 'can' (noun) is hardly a stopword in the context of `http://dmoz.org/Science/Environment/Pollution_Prevention_and_Recycling/`, which may specialize into subtopics related to cans, bottles, paper, etc. (and therefore 'can' might be an excellent feature at this level). Thus, stopwords are best determined by statistical testing on the corpus at hand w.r.t. the current topic context.

Various techniques can be used to grade terms on their "discriminating power" [72]. One option is to use mutual

[11] `http://www.w3.org/XML/`

[12] `http://www-db.stanford.edu/~widom/xml-whitepaper.html`

[13] `http://www.socks.nec.com/`

information [19]. Another option is to use class-preserving projections such as the Fisher index [27].

Ranking the terms gives some guidance on which to retain in T and which to throw out as 'noise.' One idea commonly used is to order the terms best first and retain a *prefix*. This option can be made near-linear in the number of initial features and I/O efficient [14]. However this static ranking does not recognize that selecting (or discarding) a term may lead to a change in the desirability score of another term. More sophisticated (and more time-consuming) feature selection techniques based on finding Markov blankets in Bayesian networks [38] have been proposed [44].

Experiments suggest that feature selection yields a significant improvement for the binary naive Bayes classifier and a moderate improvement for the multinomial naive Bayes classifier [44, 72, 13, 14, 51]. Performing some form of feature selection is important; the results are not too sensitive to the exact term scoring measures used.

3.1.3 Limited dependence modeling

The most general model of term dependence that can be used in this context is a Bayesian network [38] of the following form: there is a single node that encodes the class of the document; edges from this node go to 'feature' terms chosen from T, these term nodes are arbitrarily connected with each other (but there are no directed cycles); and there are no other edges in the Bayesian network. Deriving the edges among T from training data is made difficult by the enormous dimensionality of the problem space. $|T|$ is usually large, in the range of tens of thousands of terms. One approach to cut down the complexity (to slightly worse than $|T|^2$) is to insist that each node in T has at most d ancestors, and pick them greedily based on pre-estimated pairwise correlation between terms [44]. As in simple greedy approaches, it is possible to discard a large fraction (say over 2/3) of T and yet maintain classification accuracy; in some cases, discarding noisy features improved accuracy by 5%. However, further experimental analyses on larger text data sets are needed to assess the impact of modeling term dependence on classification.

3.1.4 Other advanced models

Two other techniques have emerged in recent research that capture term dependence. The maximum entropy technique regards individual term occurrence rates as marginal probabilities that constrain the cell probabilities of a gigantic contingency table potentially involving any subset of terms. Operationally there are similarities with Bayesian networks in the sense that one has to choose which regions of this rather large contingency table to estimate for subsequent use in classification. In experiments, improvements in accuracy over naive Bayes have been varied and dependent on specific data sets [57].

Thus far we have been discussing distribution-based classifiers which characterize class-conditional term distributions to estimate likely generators for test documents. Another approach is to use separator-based classifiers that learn separating surfaces between the classes. Decision trees and perceptrons are examples of separator-based classifiers. Support vector machines (SVMs) [69] have yielded some of the best accuracy to date. SVMs seek to find a separator surface between classes such that a band of maximum possible thickness of the region (also called the margin) around the separator is empty, i.e., has no training points. This leads to better generalization [59, 42].

3.1.5 Hierarchies over class labels

In the traditional warehouse mining domain, class labels are drawn from a small discrete set, e.g., "fraudulent credit card transaction" vs. "normal transaction." Sometimes a mild ordering between the labels is possible, e.g., "low risk," "medium risk" and "high risk" patients. Topic directories such as Yahoo! provide a more complex scenario: the class labels have a large hierarchy defined on them, with the common interpretation that if c `is-a` c_0, all training documents belonging to c are also examples of c_0. (The Yahoo! is a directed acyclic graph, not a tree; but we restrict our discussion to trees only for simplicity.) The role of the classifier in this setting is open to some discussion; a common goal is to find the leaf node in the class hierarchy which has the highest posterior probability.

The common formulation in this setting is that $\Pr(\text{root}|d) = 1$ for any document d, and if c_0 is the parent of $c_1, \ldots, c_m$, then $\Pr(c_0|d) = \sum_i \Pr(c_i|d)$. Using the chain rule, $\Pr(c_i|d) = \Pr(c_0|d)\Pr(c_i|c_0, d)$, one starts at the root and computes posterior probabilities for all classes in the hierarchy via a best-first search, always expanding that c with the highest $\Pr(c|d)$ and collecting unexpanded nodes in a priority queue until sufficiently many leaf classes are obtained. An approximation to this is the 'Pachinko machine' [45] which, starting at the root, selects in each step the most likely child of the current class until it reaches a leaf.

Another issue that arises with a taxonomy is that of context-dependent feature selection. As mentioned before in §*3.1.2*, the set of features useful for discrimination between classes may vary significantly, depending on c_0. This has been observed in a set of experiments [13, 45] across different data sets. Surprisingly few terms (tens to hundreds) suffice at each node in the topic hierarchy to build models which are as good (or slightly better) as using the whole lexicon. Bayesian classifiers in this domain tend to have high bias and rarely overfit the data; they are quite robust to moderate excesses in parameters. Therefore the major benefit is in storage cost and speed [14]. For a data set such as Yahoo!, feature selection may make the difference between being able to hold the classifier model data in memory or not.

Yet another utility of a class hierarchy is in improved model estimation. Sparsity of data is always a problem in the text domain, especially for rare classes. Information gleaned indirectly from the class models of better populated ancestors and siblings in the taxonomy may be used to 'patch' local term distributions. One such approach is **shrinkage** [53]. Given a topic tree hierarchy, its root is made the only child of a special 'class' c_0 where all terms are equally likely ($\theta_{c_0,t} = 1/T$ for all $t \in T$). Consider a path $c_0, c_1, \ldots, c_k$ up to a leaf c_k and suppose we need to estimate the θ-parameters of the classes on this path. The set of training documents D_{c_k} for leaf c_k is used as is. For c_{k-1}, the training set used is $D_{c_{k-1}} \setminus D_{c_k}$, i.e., documents in D_{c_k} are discarded so that the estimates become independent. This process is continued up the path. From the training data, the maximum likelihood estimates of $\theta_{c,t}$ are computed; this is similar to equation (3), after dropping the '1+' and '$|T|$+' terms. Shrinkage involves estimating a

weight vector $\Lambda = (\lambda_0, \ldots, \lambda_k)$ such that the 'corrected' θ parameters for c_k are

$$\tilde{\theta}_{c_k,t} = \sum_{0 \le i \le k} \lambda_i \theta_{c_i,t}. \quad (5)$$

Λ is estimated empirically by maximizing the likelihood of a held-out portion of the training data using an Expectation Maximization (EM) framework [24] (also see §5.1).

3.2 Learning relations

Classifying documents into topics can be extended into learning relations between pages. For example, in trying to classify pages at a college into faculty, student, project, and course pages, one may wish to learn relations like

```
teaches(faculty, course),
advises(faculty, student),
enrolled(student, course),
```

etc. In turn, learning such relations may improve the accuracy of node classification.

Learning relations between hypertext pages involve a combination of statistical and relational learning. The former component can analyze textual features while the latter can exploit hyperlinks. The idea is to augment FOIL-based learner [50] with the ability to invent predicates, using a naive Bayes text classifier. Due to its relational component, it can represent hyperlink graph structure and the word statistics of the neighbouring documents. At the same time, using a statistical method for text implies that the learned rules will not be dependent on the presence or absence of specific keywords as would be the case with a purely relational method [50].

4 Unsupervised learning

In unsupervised learning [41] of hypertext, the learner is given a set of hypertext documents, and is expected to discover a hierarchy among the documents based on some notion of similarity, and organize the documents along that hierarchy. A good clustering will collect similar documents together near the leaf levels of the hierarchy and defer merging dissimilar subsets until near the root of the hierarchy. Clustering is used to enhance search, browsing, and visualization [37].

4.1 Basic clustering techniques

Clustering is a fundamental operation in structured data domains, and has been intensely studied. Some of the existing techniques, such as k-means [41] and hierarchical agglomerative clustering [21] can be applied to documents. Typically, documents are represented in unweighted or TFIDF vector space, and the similarity between two documents is the cosine of the angle between their corresponding vectors, or the distance between the vectors, provided their lengths are normalized.

4.1.1 k-means clustering

The number k of clusters desired is input to the k-means algorithm, which then picks k 'seed documents' whose coordinates in vector space are initially set arbitrarily. Iteratively, each input document is 'assigned to' the most similar seed. The coordinate of the seed in the vector space is recomputed to be the centroid of all the documents assigned to that seed. This process is repeated until the seed coordinates stabilize.

The extreme high dimensionality of text creates two problems for the top-down k-means approach. Even if each of 30000 dimensions has only two possible values, the number of input documents will always be too small to populate each of the 2^{30000} possible cells in the vector space. Hence most dimensions are unreliable for similarity computations. But since this is not a supervised problem, it is hard to detect them a priori. There exist bottom-up techniques to determine orthogonal subspaces of the original space (obtained by projecting out other dimensions) such that the clustering in those subspaces can be characterized as 'strong' [1]. However these techniques cannot deal with the tens of thousands of dimensions.

4.1.2 Agglomerative clustering

In agglomerative or bottom-up clustering, documents are continually merged into super-documents or groups until only one group is left; the merge sequence generates a hierarchy based on similarity. In one variant [21], the *self-similarity* of a group Γ is defined as

$$s(\Gamma) = \frac{1}{|\Gamma|(|\Gamma|-1)} \sum_{d_1,d_2 \in \Gamma, d_1 \neq d_2} s(d_1, d_2), \quad (6)$$

where $s(d_1, d_2)$ is the similarity between documents d_1 and d_2, often defined as the cosine of the angle between the vectors corresponding to these documents in TFIDF vector space. (Many other measures of similarity have been used.) The algorithm initially places each document into a group by itself. While there is more than one group the algorithm looks for a Γ and a Δ such as $s(\Gamma \cup \Delta)$ is maximized, and merges these two groups. This algorithm takes time that is worse than quadratic in the number of initial documents. To scale it up to large collections, a variety of sampling techniques exist [20].

4.2 Techniques from linear algebra

The vector space model and related representations suggest that linear transformations to documents and terms, regarded as vectors in Euclidean space, may expose interesting structure. We now discuss some applications of linear algebra to text analysis.

4.2.1 Latent semantic indexing

The similarity between two documents in the vector space model is a syntactic definition involving those two documents alone. (In TFIDF weighted vector space other documents do exert some influence through IDF.) However, indirect evidence often lets us build semantic connections between documents that may not even share terms. For example, 'car' and 'auto' co-occurring in a document may lead us to believe they are related. This may help us relate a document mentioning 'car' and 'gearbox' with another document mentioning 'auto' and 'transmission,' which may in turn lead us to believe 'gearbox' and 'transmission' are related.

Latent Semantic Indexing (LSI) [23] formalizes this intuition using notions from linear algebra. Consider the term-by-document matrix A where a_{ij} is 0 if term i does not occur in document j, and 1 otherwise (word counts and TFIDF weighting may also be used). Now if 'car' and 'auto' are related, we would expect them to occur in similar sets of

documents, hence the rows corresponding to these words should have some similarity. Extending this intuition we might suspect that A may have a rank r far lower than its dimensions; many rows and/or columns may be somewhat 'redundant.'

Let the singular values [34] of A (the eigenvalues of AA^T) be $\sigma_1, \ldots, \sigma_r$, where $|\sigma_1| \geq \cdots \geq |\sigma_r|$. The Singular Value Decomposition (SVD) [34] factorizes A into three matrices UDV^T, where $D = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$, and U and V are column-orthogonal matrices. LSI retains only some first k singular values together with the corresponding rows of U and V, which induce an approximation to A, denoted $A_k = U_k D_k V_k^T$ (k is a tuned parameter). Each row of U_k represents a term as a k-dimensional vector; similarly each row of V_k represents a document as a k-dimensional vector.

Among many things, this modified representation makes for a new search method: map a keyword query to a k-dimensional vector, and find documents which are nearby. (For multi-word queries one may take the centroid of the word vectors as the query.) Heuristically speaking, the hope is that the rows in U corresponding to 'car' and 'auto' will look very similar, and a query on 'car' *will* have a nonzero match with a document containing the word 'auto.' Improvement in query performance has indeed been observed [23].

In the Signal Processing community, SVD and truncation has been used for decades for robust model fitting and noise reduction [56]. Analogously for text, it has been shown that LSI extracts semantic clusters in spite of noisy mappings from clusters to terms [58]. Apart from search, LSI can also be used to preprocess the data (eliminating noise) for clustering and visualization of document collection in very low-dimensional spaces (2- or 3-d).

4.2.2 *Random projections*

In either k-means or agglomerative clustering, a significant fraction of the running time is spent in computing similarities between documents and document groups. Although individual documents can be expected to have bounded length, cluster centroids and document groups become dense as they collect contributions from more and more documents during the execution of these algorithms.

There is a theorem which establishes that a projection of a set of points to a randomly oriented subspace induces small distortions in most inter-point distances with high probability. More precisely, let $v \in \mathbf{R}^n$ be a unit vector and H a randomly oriented ℓ-dimensional subspace through the origin, and let X be the square of the length of the projection of v on to H (X is thus a random variable). Then $E[X] = \ell/n$ and if ℓ is chosen between $\Omega(\log n)$ and $O(\sqrt{n})$, $\Pr(|X - \ell/n| > \epsilon\ell/n) < 2\sqrt{\ell}\exp(-(\ell-1)\epsilon^2/4)$, where $0 < \epsilon < 1/2$ [30]. It is easy to see that this implies small distortion in inter-point distances and inner products.

This theorem can be used to develop a technique for reducing the dimensionality of the points so as to speed up the distance computation inherent in clustering programs. It has been proven [58] that the quality of the resulting clustering is not significantly worse than a clustering derived from the points in the original space. In practice, simpler methods such as truncating the document or cluster centroid vectors to retain only the most frequent terms (i.e., the largest orthogonal components) perform almost as well as projection [20, 66].

5 Semi-supervised learning

Supervised learning is a goal-directed activity which can be precisely evaluated, whereas unsupervised learning is open to interpretation. On the other hand, supervised learning needs training data which must be obtained through human effort. In real life, most often one has a relatively small collection of labeled training documents, but a larger pool of unlabeled documents.

5.1 Learning from labeled and unlabeled documents

Clearly, term sharing and similarity between labeled and unlabeled documents is a source of information that may lead to increased classification accuracy. This has indeed been verified using a simple algorithm patterned after Expectation Maximization (EM) [24]. The algorithm first trains a naive Bayes classifier using only labeled data. Thereafter each EM iteration consists of the E-step and M-step. We restrict the discussion to naive Bayesian classifiers. In the E-step one estimates $\theta_{c,t}$ using a variant of equation (3):

$$\theta_{c,t} = \frac{1 + \sum_d n(d,t)\ \Pr(c|d)}{|T| + \sum_d \Pr(c|d) \sum_\tau n(d,\tau)}. \qquad (7)$$

The modification is that documents do not belong deterministically to a single class, but may belong probabilistically to many classes. (This distribution will be degenerate for the initial labeled documents.) In the M-step, the $\theta_{c,t}$ estimates are used to assign class probabilities $\Pr(c|d)$ to all documents not labeled as part of th input. These EM iterations continue until (near-) convergence. Results are mostly favorable compared to naive Bayes alone: error is reduced by a third in the best cases, but care needs to be taken in modeling classes as mixtures of term distributions.

5.2 Relaxation labeling

In supervised topic learning from text, distribution-based classifiers posit a class-conditional term distribution $\Pr(\vec{t}|c)$ and use this to estimate the posterior class probabilities $\Pr(c|d)$ for a given document. Consider now the hypertext model in §2.2, where training and testing documents are nodes in a hypertext graph. It seems obvious that apart from terms in the training and testing documents, there are other sources of information induced by the links, but it is unclear how to exploit them.

One can adopt the notion, motivated by spreading activation techniques, that a term 'diffuses' to documents in the local link neighborhood of the document where it belongs; this has been proved useful in hypertext IR [64]. In our context, it means upgrading our model to a class-conditional distribution over not only terms in the current document d, but also neighboring documents $N(d)$ (some suitable radius can be chosen to determine $N(d)$), viz., $\Pr(\vec{t}(d), \vec{t}(N(d))|c)$. Operationally, it means that terms in neighbors of a training document d_1 contribute to the model for the class of d_1, and terms in neighbors of testing document d_2 may be used to estimate the class probabilities for d_2. (One may apply a heuristic damping function which limits the radius of influence of a term.)

Interestingly, this does not lead to better accuracy in experiments [16]. The reason, in hindsight, is that content expansion is not the only purpose of citation; in other words, it is not always the case that with respect to a set of top-

ics, hypertext forms very tightly isolated clusters. In the US Patents database, for example, one often finds citations from patents from the class 'amplifiers' to the class 'oscillators' (but none to 'pesticides'). Thus one is led to refine the model of linkages to posit a class conditional joint distribution on terms in a document and the *classes* of neighbors, denoted by $\Pr(\vec{t}(d), \vec{c}(N(d))|c)$. Note that there is a circularity involving class labels.

One approach to resolve the circularity is to initially use a text-only classifier to assign initial class probabilities $\Pr_{(0)}(c|d)$ to all documents in a suitable neighborhood of the test document, then iteratively compute, for each node at step $r+1$,

$$\Pr_{(r+1)}(c|d, N(d)) = \sum_{\vec{c}(N(d))} \Pr_{(r)}\vec{c}(N(d)) \Pr_{(r)}(c|d, \vec{c}(N(d))), \quad (8)$$

where the sum is over all possible neighborhood class assignments. (In practice one truncates this sum.)

In the case where some of the nodes in $N(d)$ are preclassified, this becomes a partially supervised learning problem. Relaxation labeling is able to smoothly improve its accuracy as this happens [16].

6 Social network analysis

The web is an example of a *social network*. Social networks have been extensively researched [70]. Social networks are formed between academics by co-authoring, advising, serving on committees; between movie personnel by directing and acting; between musicians, football stars, friends and relatives; between people by making phone calls and transmitting infection; between papers through citation, and between web pages by hyperlinking to other web pages.

Social network theory is concerned with properties related to connectivity and distances in graphs, with diverse applications like epidemiology, espionage, citation indexing, etc. In the first two examples, one might be interested in identifying a few nodes to be removed to significantly increase average path length between pairs of nodes. In citation analysis, one may wish to identify influential or central papers; this turns out to be quite symmetric to finding good survey papers; this symmetry has been explored by Mizruchi and others [55]. IR literature includes insightful studies of citation, co-citation, and influence of academic publication [48].

6.1 Applying social network analysis to the Web

Starting in 1996, a series of applications of social network analysis were made to the web graph, with the purpose of identifying the most authoritative pages related to a user query.

6.1.1 Google

If one wanders on the Web for infinite time, following a random link out of each page with probability $1-p$ and jumps to a random Web page with probability p, then different pages will be visited at different rates; popular pages with many in-links will tend to be visited more often. This measure of popularity is called PageRank [10], defined recursively as

$$\text{PageRank}(v) = p/N + (1-p)\sum_{u \to v} \frac{\text{PageRank}(u)}{\text{OutDegree}(u)}, \quad (9)$$

where '$\to$' means "links to" and N is the total number of nodes in the Web graph. (The artifice of p is needed because the Web is not connected or known to be aperiodic, therefore the simpler eigenequation is not guaranteed to have a fixed point.) The **Google**[14] search engine simulates such a random walk on the web graph in order to estimate PageRank, which is used as a score of popularity. Given a keyword query, matching documents are ordered by this score. Note that the popularity score is precomputed independent of the query, hence Google can be potentially as fast as any relevance-ranking search engine.

6.1.2 Hyperlink induced topic search (HITS)

Hyperlink induced topic search [43] is slightly different: it does not crawl or pre-process the web, but depends on a search engine. A query to HITS is forwarded to a search engine such as Alta Vista, which retrieves a subgraph of the web whose nodes (pages) match the query. Pages citing or cited by these pages are also included. Each node u in this expanded graph has two associated scores h_u and a_u, initialized to 1. HITS then iteratively assigns

$$a_v := \sum_{u \to v} h_u \quad \text{and} \quad h_u := \sum_{u \to v} a_v, \quad (10)$$

where $\sum_u h_u$ and $\sum_v a_v$ are normalized to 1 after each iteration. The a and h scores converge respectively to the measure of a page being an authority, and the measure of a page being a *hub* (a compilation of links to authorities, or a "survey paper" in bibliometric terms).

Because of the query-dependent graph construction, HITS is slower than Google. A variant of this technique has been used by Dean and Henzinger to find similar pages on the Web using link-based analysis alone [22]. They improve speed by fetching the Web graph from a *connectivity server* which has pre-crawled substantial portions of the Web [7].

6.1.3 Adding text information to link-based popularity

HITS's graph expansion sometimes leads to topic *contamination* or *drift*. E.g., the community of *movie awards* pages on the web is closely knit with highly cited (and to some extent relevant) home pages of movie *companies*. Although *movie awards* is a finer topic than *movies*, the top movie companies emerge as the victors upon running HITS. This is partly because in HITS (and Google) all edges in the graph have the same importance[15]. Contamination can be reduced by recognizing that hyperlinks that contain *award* or *awards* near the anchor text are more relevant for this query than other edges. The Automatic Resource Compilation (ARC) and Clever systems incorporate such query-dependent modification of edge weights [15, 17]. Query results are significantly improved. In user studies, the results compared favorably with lists compiled by humans, such as Yahoo! and **Infoseek**[16].

6.1.4 Outlier filtering

Bharat and Henzinger have invented another way to integrate textual content and thereby avoid contamination of

[14] `http://google.com`

[15] Google might be using edge-weighting strategies which are not published.

[16] `http://www.infoseek.com`

the graph to be distilled. They model each page according to the "vector space" model [63]. During the graph expansion step, unlike HITS, they do not include all nodes at distance one from the preliminary query result. Instead they prune the graph expansion at nodes whose corresponding term vectors are outliers with respect to the set of vectors corresponding to documents directly retrieved from the search engine [8]. In the example above, one would hope that the response to the query `movie award` from the initial keyword search would contain a majority of pages related to awards and not companies; thus the distribution of keywords on these pages will enable the Bharat and Henzinger algorithm to effectively prune away as outliers nodes in the neighborhood that are about movie companies. Apart from speeding up their system by using the Connectivity Server [7], they describe several heuristics that cut down the query time substantially.

It is possible to fabricate queries that demonstrate the strengths and weaknesses of each of these systems. ARC, Clever, and Outlier Filtering have been shown to be better (as judged by testers) than HITS. There has not been a systematic comparison between Google and the HITS family. This would be of great interest given the basic difference in graph construction and consequent greater speed of Google.

6.2 Resource discovery

There is short-range topic locality on the Web, in at least two prominent forms. If the reader finds this paper interesting, there is reasonable chance that the reader will also find some significant fraction of citations in this paper to be interesting too. If this paper cites a paper that the reader thinks is interesting, that makes it more likely that another paper cited by this one is interesting (i.e., this paper is a suitable *hub*).

Coupling the ability to find good hubs with the ability to judge how likely a given page is to be generated from a given topic of interest, one can devise crawlers that crawl the Web for selected topics [18]. The goal is to crawl as many relevant pages as fast as possible while crawling as few irrelevant pages as possible. The crawler can be guided by a text or hypertext classifier. Alternatively, one may use reinforcement learning techniques [60, 52].

7 Conclusion

In this survey we have reviewed recent research on the application of techniques from machine learning, statistical pattern recognition, and data mining to analyzing hypertext. Starting from simple bulk models for documents and hyperlinks, researchers have progressed towards document substructure and the interplay between link and content. Mining of semistructured data has followed a slightly different path, but it seems clear that a confluence will be valuable. Commercial search products and services are slowly adopting the results of recent investigations. It seems inevitable that knowledge bases and robust algorithms from computational linguistics will play an increasingly important role in hypertext content management and mining in the future. We will also likely see increased systems and tool building as the winning research ideas become more firmly established.

Acknowledgements: Thanks to Pushpak Bhattacharya and Yusuf Batterywala for helpful discussions and references.

8 REFERENCES

[1] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *SIGMOD Conference on Management of Data*, Seattle, June 1998. Online at `http://www.almaden.ibm.com/cs/quest/papers/sigmod98_clique.pdf`.

[2] J. Allan. Automatic hypertext link typing. In *7th ACM Conference on Hypertext, Hypertext '96*, pages 42–51, 1996.

[3] J. Allen. *Natural Language Understanding.* Benjamin/Cummings, 1987.

[4] D. J. Arnold, L. Balkan, R. L. Humphreys, S. Meijer, and L. Sadler. Machine translation: An introductory guide, 1995. Online at `http://clwww.essex.ac.uk/~doug/book/book.html`.

[5] *Babelfish Language Translation Service.* `http://www.altavista.com`, 1998.

[6] K. Bharat and A. Broder. A technique for measuring the relative size and overlap of public web search engines. In *7th World-Wide Web Conference (WWW7)*, 1998. Online at `http://www7.scu.edu.au/programme/fullpapers/1937/com1937.htm`; also see an update at `http://www.research.digital.com/SRC/whatsnew/sem.html`.

[7] K. Bharat, A. Bröder, M. Henzinger, P. Kumar, and S. Venkatasubramanian. The Connectivity Server: Fast access to linkage information on the Web. In *7th World Wide Web Conference*, Brisbane, Australia, 1998. Online at `http://www.research.digital.com/SRC/personal/Andrei_Broder/cserv/386.ht%ml` and `http://decweb.ethz.ch/WWW7/1938/com1938.htm`.

[8] K. Bharat and M. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *21st International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 104–111, Aug. 1998. Online at `ftp://ftp.digital.com/pub/DEC/SRC/publications/monika/sigir98.pdf`.

[9] W. J. Bluestein. Hypertext versions of journal articles: Computer aided linking and realistic human evaluation. In *PhD Thesis, University of Western Ontario*, 1999.

[10] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the 7th World-Wide Web Conference (WWW7)*, 1998. Online at `http://decweb.ethz.ch/WWW7/1921/com1921.htm`.

[11] C. Cardie and D. Pierce. The role of lexicalization and pruning for base noun phrase grammars. In *AAA1 99*, pages 423–430, July 1999.

[12] N. Catenazzi and F. Gibb. The publishing process: the hyperbook approach. *Journal of Information Science*, 21(3):161–172, 1995.

[13] S. Chakrabarti, B. Dom, R. Agrawal, and P. Raghavan. Using taxonomy, discriminants, and signatures to navigate in text databases. In *VLDB*, Athens, Greece, Sept. 1997.

[14] S. Chakrabarti, B. Dom, R. Agrawal, and P. Raghavan. Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies. *VLDB Journal*, Aug. 1998. Invited paper, online at `http://www.cs.berkeley.edu/~soumen/VLDB54_3.PDF`.

[15] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan. Automatic resource compilation by analyzing hyperlink structure and associated text. In *7th World-wide web conference (WWW7)*, 1998. Online at `http://www7.scu.edu.au/programme/fullpapers/1898/com1898.html`.

[16] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *SIGMOD Conference*. ACM, 1998. Online at `http://www.cs.berkeley.edu/~soumen/sigmod98.ps`.

[17] S. Chakrabarti, B. E. Dom, S. Ravi Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, D. Gibson, and J. Kleinberg. Mining the Web's link structure. *IEEE Computer*, 32(8):60–67, Aug. 1999. Feature article.

[18] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: a new approach to topic-specific web resource discovery. *Computer Networks*, 31:1623–1640, 1999. First appeared in the **8th International World Wide Web Conference**[17], Toronto, May 1999. Available online at `http://www8.org/w8-papers/5a-search-query/crawling/index.html`.

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

[20] D. R. Cutting, D. R. Karger, and J. O. Pedersen. Constant interaction-time scatter/gather browsing of very large document collections. In *Annual International Conference on Research and Development in Information Retrieval*, 1993.

[21] D. R. Cutting, D. R. Karger, J. O. Pedersen, and J. W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *Annual International Conference on Research and Development in Information Retrieval*, Denmark, 1992.

[22] J. Dean and M. R. Henzinger. Finding related pages in the world wide web. In *8th World Wide Web Conference*, Toronto, May 1999.

[23] 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. Online at `http://superbook.telcordia.com/~remde/lsi/papers/JASIS90.ps`.

[24] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, B(39):1–38, 1977.

[25] S. DeRose. What do those weird XML types want, anyway? Keynote address, VLDB 1999, Edinburgh, Scotland, Sept. 1999.

[26] P. Domingos and M. Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29:103–130, 1997.

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

[28] S. Fong and R. Berwick. *Parsing with Principles and Parameters*. MIT Press, 1992.

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

[30] P. Frankl and H. Maehara. The Johnson-Lindenstrauss lemma and the sphericity of some graphs. *Journal of Combinatorial Theory*, B 44:355–362, 1988.

[31] J. H. Friedman. On bias, variance, 0/1 loss, and the curse of dimensionality. *Data Mining and Knowledge Discovery*, 1(1):55–77, 1997. Stanford University Technical Report, online at `ftp://playfair.stanford.edu/pub/friedman/curse.ps.Z`.

[32] G. Gazder and C. Mellish. *Natural Language Processing in LISP*. Addison-Wesley, 1989.

[33] R. Goldman, J. McHugh, and J. Widom. From semistructured data to XML: Migrating the Lore data model and query language. In *Proceedings of the 2nd International Workshop on the Web and Databases (WebDB '99)*, pages 25–30, Philadelphia, June 1999. Online at `http://www-db.stanford.edu/pub/papers/xml.ps`.

[34] G. H. Golub and C. F. van Loan. *Matrix Computations*. Johns Hopkins University Press, London, 1989.

[35] S. G. Green. Building newspaper links in newspaper articles using semantic similarity. In *Natural Language and Data Bases Conference, Vancouver, NLDB'97*, 1997.

[36] L. Haegeman. *Introduction to Government and Binding Theory*. Basil Blackwell Ltd., Oxford, 1991.

[37] 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 the 20th Annual International ACM/SIGIR Conference*, Philadelphia, PA, July 1997. Online at `ftp://parcftp.xerox.com/pub/hearst/sigir97.ps`.

[17] `http://www8.org`

[38] D. Heckerman. Bayesian networds for data mining. *Data Mining and Knowledge Discovery*, 1(1), 1997. Also see URLs `ftp://ftp.research.microsoft.com/pub/dtg/david/tutorial.PS` and `ftp://ftp.research.microsoft.com/pub/tr/TR-95-06.PS`.

[39] W. J. Hutchins and H. L. Somers. *An Introduction to Machine Translation*. Academic Press, 1992.

[40] U. N. U. Institute of Advanced Studies. The universal networking language: Specification document. In *Internal Technical Document*, 1999.

[41] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[42] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*. MIT Press, 1999. Also see `http://www-ai.informatik.uni-dortmund.de/FORSCHUNG/VERFAHREN/SVM_LIGHT/%svm_light.eng.html`.

[43] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *ACM-SIAM Symposium on Discrete Algorithms*, 1998. Online at `http://www.cs.cornell.edu/home/kleinber/auth.ps`.

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

[45] D. Koller and M. Sahami. Hierarchically classifying documents using very few words. In *International Conference on Machine Learning*, volume 14. Morgan-Kaufmann, July 1997. Online at `http://robotics.stanford.edu/users/sahami/papers-dir/ml97-hier.ps`.

[46] K. S. Kumarvel. Automatic hypertext creation. *M.Tech Thesis, Computer Science and Engineering Department, IIT Bombay*, 1997.

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

[48] R. Larson. Bibliometrics of the world wide web: An exploratory analysis of the intellectual structure of cyberspace. In *Annual Meeting of the American Society for Information Science*, 1996. Online at `http://sherlock.berkeley.edu/asis96/asis96.html`.

[49] S. Lawrence and C. Lee Giles. Accessibility of information on the web. *Nature*, 400:107–109, July 1999.

[50] S. S. Mark Craven and K. Nigam. First order learning for web mining. *10th European Conference on Machine Learning*, 1998.

[51] A. McCallum and K. Nigam. A comparison of event models for naive Bayes text classification. In *AAAI/ICML-98 Workshop on Learning for Text Categorization*, pages 41–48. AAAI Press, 1998. Also technical report WS-98-05, CMU; online at `http://www.cs.cmu.edu/~knigam/papers/multinomial-aaaiws98.pdf`.

[52] A. McCallum, K. Nigam, J. Rennie, and K. Seymore. Building domain-specific search engines with machine learning techniques. In *AAAI-99 Spring Symposium*, 1999. Online at `http://www.cs.cmu.edu/~mccallum/papers/cora-aaaiss99.ps.gz`.

[53] A. McCallum, R. Rosenfeld, T. Mitchell, and A. Ng. Improving text classification by shrinkage in a hierarchy of classes. In *ICML*, 1998. Online at `http://www.cs.cmu.edu/~mccallum/papers/hier-icml98.ps.gz`.

[54] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom. Lore: A database management system for semistructured data. *SIGMOD Record*, 26(3):54–66, Sept. 1997. Online at `http://www-db.stanford.edu/pub/papers/lore97.ps`.

[55] M. S. Mizruchi, P. Mariolis, M. Schwartz, and B. Mintz. Techniques for disaggregating centrality scores in social networks. In N. B. Tuma, editor, *Sociological Methodology*, pages 26–48. Jossey-Bass, San Francisco, 1986.

[56] T. K. Moon and W. C. Sterling. *Mathematical Methods and Algorithms for Signal Processing*. Prentice Hall, 1 edition, Aug. 1999.

[57] K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification. In *IJCAI'99 Workshop on Information Filtering*, 1999. Online at `http://www.cs.cmu.edu/~mccallum/papers/maxent-ijcaiws99.ps.gz`.

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

[59] J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14, Microsoft Research, 1998. Online at `http://www.research.microsoft.com/users/jplatt/smoTR.pdf`.

[60] J. Rennie and A. McCallum. Using reinforcement learning to spider the web efficiently. In *ICML*, 1999. Online at `http://www.cs.cmu.edu/~mccallum/papers/rlspider-icml99s.ps.gz`.

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

[62] G. Salton. *Automatic Text Processing*. Addison Wesley, 1989.

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

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

[65] R. G. Schank and C. J. Rieger. Inference and computer understanding of natural language. In *in Readings in Knowledge Representation, R.J. Brachman and H.J.Levesque (ed.), Morgan Kaufmann Publishers*, 1985.

[66] H. Schütze and C. Silverstein. A comparison of projections for efficient document clustering. In *Proceedings of the Twentieth Annual ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 74–81, July 1997. Online at `http://www-cs-students.stanford.edu/~csilvers/papers/metrics-sigir.ps`.

[67] J. F. Sowa. *Conceptual Structures: Information Processing in Mind and Machines.* Addison Wesley, 1984.

[68] D. Temperley. An introduction to link grammar parser. Technical report, Apr. 1999. Online at `http://www.link.cs.cmu.edu/link/dict/introduction.html`.

[69] V. Vapnik, S. Golowich, and A. J. Smola. Support vector method for function approximation, regression estimation, and signal processing. In *Advances in Neural Information Processing Systems.* MIT Press, 1996.

[70] S. Wasserman and K. Faust. *Social Network Analysis.* Cambridge University Press, 1994.

[71] T. Winograd. *Language as a Cognitive Process, Volume 1: Syntax.* Addison-Wesley, Reading, MA, 1983.

[72] Y. Yang and J. Pedersen. A comparative study on feature selection in text categorization. In *International Conference on Machine Learning*, pages 412–420, 1997.

Biography: **Soumen Chakrabarti**[18] received his B.Tech in Computer Science from the Indian Institute of Technology, Kharagpur, in 1991 and his M.S. and Ph.D. in Computer Science from the University of California, Berkeley in 1992 and 1996. At Berkeley he worked on compilers and runtime systems for running scalable parallel scientific software on message passing multiprocessors. He was a Research Staff Member at IBM Almaden Research Center between 1996 and 1999. At IBM he worked on hypertext analysis and information retrieval. He designed the **Focused Crawler**[19] and part of the **Clever**[20] search engine. He is currently an Assistant Professor in the Department of Computer Science and Engineering at the Indian Institute of Technology, Bombay. His research interests include hypertext information retrieval, web analysis and data mining.

[18] `http://www.cse.iitb.ernet.in/~soumen/`
[19] `http://www.cs.berkeley.edu/~soumen/focus`
[20] `http://www.almaden.ibm.com/cs/k53/clever.html`