Monitoring Message Streams: Algorithmic Methods for Automatic Processing of Messages

April 27, 2003

Paul B. Kantor and Fred S. Roberts

Rutgers University, New Brunswick, NJ 08903

Abstract

The problem of monitoring message streams is decomposed into problems in compression, representation, matching, learning and data fusion. A coordinated approach aimed at improvements in each of the several components is described. Exploration of the space of possible combinations of approaches, and of fusion of multiple methods for each of the component problems is aimed at order of magnitude improvements in overall performance. Initial investigations are reported, with pointers to more detailed presentations of the methods and results.

1 Introduction and Objective of Project

We describe a multidisciplinary project being carried out at DIMACS, the Center for Discrete Mathematics and Theoretical Computer Science, based at Rutgers University. The project seeks to advance the state of the art for filtering streams of text messages, and identifying those that are relevant to a given (already identified) topic or event.

DIMACS is a consortium of academic and industrial institutions (Rutgers and Princeton Universities, AT&T Labs, Bell Labs (Lucent Technologies), NEC Laboratories America, and Telcordia Technologies, with affiliated partners at Avaya Labs, IBM Research, and Microsoft Research). The project has included researchers from Rutgers, AT&T Labs, Avaya Labs, and Telcordia Technologies. The diversity of resources and of analytical perspectives available to us is a key to the success of the project.

This paper is organized as follows. Section 1 lays out the general analytic framework. Section 2 discusses the infrastructure that we have built to address the research issues. Section 3 sketches a number of approaches to dimensionality reduction. In Section 4, we discuss problems of feature selection. Section 5 explores our study of Bayesian classifiers appropriate to this problem. An exploration of fusion of methods for matching documents to profiles or categories is the subject of Section 6 while the general problem of streaming data analysis is the subject of Section 7. Section 8 sketches two approaches to the problem of building a formal model for the monitoring problem and Section 9 summarizes the present state of the project and outlines future work.

1.1 The Problem

The project seeks to improve existing methods for monitoring huge streams of textualized communications to automatically identify clusters of messages relating to significant "events." We are studying retrospective or "supervised" event identification, where the system begins with a set of documents that have been associated with one or more events called "labeled documents." The task is to automatically associate a relevant new message with one or more of these pre-existing events. This task is of course of great value in forensic activities. In later years, we will also study prospective detection (or "discovery" or "unsupervised learning"), where we must decide whether a "new event" is present in the flow of messages.

1.2 Multi-Disciplinary Perspective

The problems involved in automatic processing of messages call for a variety of skills and backgrounds. We have brought together a team of statisticians, computer scientists, and experts in information retrieval and library science, consisting of nine Ph.D. researchers, two graduate student assistants, and two programmers. The team includes an expert in information retrieval, with extensive experience in designing and evaluating operational text classification systems; an expert on statistical methods for dealing with massive data sets who has developed powerful extensions of basic Bayes classifiers; and an information scientist whose work on combining multiple methods for classifying or ranking candidate documents has demonstrated performance superior to that achievable by any one system alone. Another member of our team was one of the pioneers in the use of kernel methods for machine learning and has developed a fast statistical clustering algorithm that can deal with millions of cases in reasonable time while still another has developed extremely useful methods for Boolean representation and rule learning. Three team members have developed fast compression methods that are useful in one pass through data. Finally, another member of our team has developed methods/guidelines for combining scores in software and hardware testing and has long-standing expertise in decision making and the social sciences.

1.3 Analytical Framework

The problem we are interested in can be discussed in terms of five interrelated "components." Our premise is that significant performance improvements in the automatic processing of messages will result from innovations involving more than one component of the process. The components are:

(1) ***compression*** of text to meet storage and processing limitations;

(2) ***representations*** of text into a form amenable to computation and statistical analysis;

(3) ***matching schemes*** for computing similarity between documents in terms of the representation chosen;

(4) ***learning methods*** building on a set of judged examples to determine the key characteristics of a document cluster or "event"; and

(5) ***fusion schemes*** that combine methods that are "sufficiently different" to yield improved detection and clustering of documents.

Of course, these distinctions are a bit arbitrary and many approaches to message processing overlap several of these components. Indeed, existing methods make use of some or all five components. However, they don't exploit the full power of the components and/or an understanding of how to apply them to text data.

Thus, the overall design of our project can be seen as exploratory experimentation, in which each of the several components is varied separately to identify which choices for each component are *individually* effective. Then choices for each component will be combined to see which are most effective when used in *combination*.

1.3.1 Compression and Dimension Reduction

In this project representation and compression become intertwined through the study of dimension reduction as a form of compression. Existing methods of compression in information retrieval are typically applied to the index structures, and aim at fitting into available computational resources without paying attention to upfront data compression. Recent research demonstrates that, for many statistical methods, dramatic reductions in space and time complexity might be realized with only a controllably small impact on the methods' numerical properties, by making a preprocessing pass over the data to reduce its dimension [100, 54, 24, 12, 53, 31]. Unlike feature-extracting dimension reduction, which, done without care, may lead to very bad results, the recent algorithmic techniques, including randomized projections, perform no semantic function themselves, and so represent a different paradigm

We believe that sophisticated dimension reduction methods in a preprocessing stage followed by the use of sophisticated statistical tools in later stages can be a very powerful approach. Our goal is to identify the best combination of such newer methods through a careful exploration of a variety of tools. We are addressing issues of effectiveness (how well the task is done) and efficiency (in computational time and space) and using a combination of new or modified algorithms and improved statistical methods built on the algorithmic primitives.

1.4 Data Sets Used

No readily-available unclassified data set has all of the characteristics of the type of data on which we expect our methods to be used. Thus, we have decided to use a number of different data sets, each reflecting different relevant characteristics, and together representative of the heterogeneity of real data. We have been using the data from the Text Retrieval Conference, ***TREC*** [93] including all routing and filtering data, and in particular used several time-stamped subsets of this data (order 10^5 to 10^6 messages) in our initial work. We are also using Reuters Corpus Volume 1 (8×10^5 messages). We also have collateral data (order 10^7 MedLine Abstracts, with human indexing), which can be used to explore the robustness of our methods across change in subject field and genre. With the array of data we have available, we have been able to develop algorithms quickly, working with the smaller sets, and then analyze and improve their scalability by moving up to the larger datasets. Because members of the project team were experienced in dealing with the TREC and Reuters and Medline data sets, there was no "learning curve" delay at the start of the project.

1.5 Representation of the Texts

Modern text classification and retrieval begins at a relatively low level of mathematical sophistication [37, 9]. At this level, a document is represented by a string of numbers. In the most general case, there is a separate number for each of the unique words appearing in the document, giving the number of times the word appears. This is sometimes extended to include the number of times that different references to the same entity (United States, U.S., America, etc.) occur. The set of numbers together are placed in a standard order and form a vector, which represents the terms occurring in the document.

Another document in which the words have been rearranged would have exactly the same vector. For this reason, vector representation of documents is sometimes referred to as a ***ba gof words*** representation. Sometimes it is useful just to determine whether or not the number of times a term appears in a document exceeds some threshold. In this case, the frequency count of a term is replaced by 1 or 0, with 1 representing the fact that it exceeds the threshold. Here, we might say that the bag of words representation is reduced to a ***bag of bits*** representation. In the simplest case, we take the threshold to be 0, i.e., we take the frequency count to be 1 if the term appears at all, and 0 otherwise. Our work has involved experimentation with both bag of words and bag of bits representations.

Vector representations can be extended beyond the bag of words concept. One approach is to ignore the distinguished role of "whitespace" and use overlapping strings of characters (sometimes called ***"n-grams"***). If these strings are long enough, they preserve information about relations among consecutive words in a text and thus preserve more information about the "meaning of the text."

However, none of these computational approaches involve linguistic considerations, or what an ordinary reader would take to be the semantics of the text. The surprising success of these (apparently very naive) elementary text classification methods traces to the observation, first made formally by Luhn [62, 63], that it is difficult to talk about specific topics (especially technical topics) without using words that are common to those topics. In other words, it should be possible to determine what a document "is about" using methods that are far too weak to determine "what the document says."

The work reported here uses the bag of words, or term frequency vector, approach, and their natural extensions to bits or strings.

1.6 IDA/CCR Workshop/Tutorial on Mining Massive Data Sets and Streams

At the project start, we organized a workshop and tutorial reviewing the state of the art in streaming message filtering. This workshop, held at the Institute for Defense Analyses/Center for Communications Research, IDA/CCR and supported by them, was entitled "Workshop/Tutorial on Mining Massive Data Sets and Streams: Mathematical Methods and

Algorithms for Homeland Defense." Details of this event can be found in the report [42] and at http://dimacs.rutgers.edu/Workshops/Homeland.

2 Infrastructure Work

We began by developing a common platform for our text filtering experiments. The work involved modifying CMU's Lemur retrieval toolkit [101] creating a newswire testset with test information needs, evaluating and modifying thresholding software, experimenting with a fundamental adaptive linear classifier (Rocchio), and benchmarking several methods.

We developed our text filtering platform around the framework of TREC, which seemed a good way to quickly set up a formal framework for our research and focus members of the team on a common problem.

Recall that the goal of filtering is to process a collection of documents, assigning those that are appropriate to each of several classes, commonly called "profiles." In the TREC setting, profiles are represented by a textual description of the topic of interest, and some number of documents that have already been judged to be relevant to that topic. In TREC, one distinguishes ***adaptive filtering*** and ***batch filtering***. Both involve what we called supervised learning in Section 1. In adaptive filtering, one must "pay" for one's "education" while in batch filtering the education is provided "gratis" before any evaluated activities begin.

Filtering methods involve two mathematical components. The first, usually called a ***scoring function***, $s(d)$, maps the set of all documents d into some ordered set (in fact, this is usually taken to be the reals). The second is a parameter τ, called the ***threshold***, which divides the ordered set into two non-overlapping parts. Documents whose score exceeds the threshold are "judged to be relevant," while the others are not. Note that the practice of setting the threshold to 0 and incorporating τ into the definition of the scoring function is not customary in information retrieval generally. In batch filtering the scoring rule and the threshold are fixed before any "test documents" are presented. In adaptive filtering both the scoring rule and the threshold may be changed at any time during the evaluation process. However, under the TREC rules, one may not revise one's decision about any document after the next document has been considered.

In the adaptive setting in TREC2002, only three relevant documents are available for initialization. In addition, a set of some 80,000 documents gives base information about the vocabulary common to the documents of interest, which is used in tuning the retrieval mechanisms. Such mechanisms generally are built using weighted sums of the frequencies with which terms occur in the document to be classified. Selection of terms is initialized by the terms that appear in the known positive documents or in the description of the topics. The weights themselves are almost always adjusted to give less weight to more common words, which are presumed to be of little discriminative value.

Some words are so common as to be useless in the classification of texts, and for these the weight is set to zero. In the interests of efficiency, this is usually achieved by removing the frequencies of those terms from the term vector at the outset. Such words are called "stop words" and are maintained in a list of stop words. Very low frequency words pose a somewhat different problem. They are either misspellings of other words, or they are proper names, which might be of great value in the classification of texts. Various approaches to the very low frequency terms have been used in the text filtering context. Typically unique words are ignored, and words occuring more than some small number of times (such as 3) are included, with the highest weight. Weighting formulas generally involve the reciprocal of a logarithm.

The concept of a ***feature***, in general, extends the notion of term frequency as described above. Technically, any number f that enters into a scoring formula such as

$$s(d) = g(f, \text{other variables})$$

is a "feature." As noted, the frequency of a term (like "nuclear") in the documents could be a feature. In fact, the terms themselves are often loosely referred to as the features. For instance if we use a bag of bits representation there is a single binary feature corresponding to each term that just tells us whether or not a term is present. Note also that *any* function of available features is also a feature. If its use provides an economy of description, or computation, we keep it.

One may describe scoring rules, quite generally, as bilinear forms which are sums over features. If $d(f)$ is the score of document d on feature f, $p(f)$ is the score of profile p on feature f, and $W(f, f')$ is a weight linking the pair of features f, f', then the score $s(d|p)$ assigned to document d with regard to profile p is

$$s(d|p) = \sum_{f,f'} d(f)W(f, f')p(f')$$

In methods that use feature selection, but not feature combination, the weight matrix $W(f, f')$ is diagonal in feature space. When features are combined to provide new features, the weight matrix (in the space of original features) will generally have off-diagonal terms as well.

Sometimes, text filtering makes use of the information gained by human evaluation of documents to define a scoring scheme. The simplest model for this process is an approach called ***Rocchio filtering*** [82]. In Rocchio filtering the vector representing the topic is modified by adding to it some multiple of each vector representing a document known to be relevant, and subtracting some other multiple of each vector representing a document known to be not relevant. The scoring function is then the inner product of this vector with the vector representing each incoming document. To use this model of text filtering, one must decide (a) the weights to be assigned to judged positive and negative examples, (b) the rules for adjusting those weights, (c) the initial threshold parameter, and (d) the rules for adjusting the threshold parameter.

One further key idea, in addition to the linear scoring function and the threshold, is to learn something from documents for which no judgments are, in fact, available. This technique, called ***pseudo-relevance feedback*** [79], simply presumes that there is some number, called the ***pseudo threshold*** τ_ψ, such that it will be advantageous to treat documents whose score is above τ_ψ as if they are known to be relevant, while those below are treated as known to be not relevant, to a given topic or profile.

We built a complete simulation of the TREC setting which is referred to here as the ***Sandbox***. This Sandbox was formed using an older body of TREC documents with the training corpus about the same size and a test corpus about 1/3 the size of the actual test corpus.

Using the Sandbox, we were able to experiment with the weights appearing in the Rocchio formula, and with a rule for setting the threshold τ_ψ. The resulting best values of the parameters were used in our runs on the actual TREC data.

The details of this process, the resulting parameters, and their performance on the TREC data are summarized in [5]. In this TREC adaptive task, incoming documents are filtered and those exceeding a threshold are compared with a list of known relevant documents to simulate human judgment. In our most successful runs, the adaptive process was further modified in one set of runs (identified by the string "30") by a non-linear process of keeping only the features corresponding to those 30 terms appearing to have the greatest discriminative power. These features (terms) were reselected after each judgment had been made.

This selection of the top 30 terms is computationally intensive, as it involves reordering the terms and computing a new threshold each time a judgment is received. Although it is linear in the most general sense (that is the space of vectors is projected successively into different selected subspaces), it is computationally intensive since the "inner products" must be recomputed. For more on identifying the "most frequent" terms in a streaming data context, see [23]. Related papers on finding all terms that appear at least a certain number of times can be found in [52, 27, 40]. At present extending those approaches to determination of the most discriminating terms remains open.

Results in the TREC evaluation showed that this "term selection approach," although naive, yielded better results than our other Rocchio filters, which did not involve term selection. This sets the stage for additional studies of feature selection, which are discussed below. We have also been able to establish that the simple Rocchio framework represents, in principle, a very good choice of baseline. One of the TREC teams, from the Chinese Academy of Science [98], used a variant of this approach to achieve the best overall score in TREC2002. We have established a homotopy (that is, a continuous path in the space of model parameters) between our initial method and that used by [98]. We do not yet know how to adjust our method of initialization to achieve those parameters *before* the evaluation phase of the TREC adaptive filtering task. Summary statistics can be found at [51].

3 Reducing Dimensionality by Preprocessing and Matching

Feature selection is the general name for any process that either selects among the existing features (for example, by dropping some terms, and the corresponding components of the document vector, as done in our "top 30" schemes) or creates new "composite features" for example by linear or non-linear transformation of the original feature values. It may be introduced to improve either classification performance or computational efficiency.

There is an interplay between intelligent selection of features and the problem of speeding up computations and/or reducing the computational resources (memory and storage) needed to perform them. Our work in this area involves a mix of the components we have called compression and matching. The results of a study such as ours are perhaps best represented by a chart in which one axis represents the speed at which computations are completed and the other represents the effectiveness of those computations - as measured by any of several standard measures of performance.

The canonical approach to computing inner products uses a data structure called an inverted file, which is an indexed set indexed by the terms that define the basis in the vector space of documents and queries. We have considered, so far, four other approaches to speeding these computations through preprocessing and matching: (1) extending inverted files to nearest neighbor calculations; (2) projection to a collection of randomly chosen real subspaces; (3) projection to a collection of randomly chosen Boolean spaces; and (4) selection of deviant cases using streaming techniques.

The first, and most conservative, approach is adapting efficiency techniques used in search engines to the filtering problem, and particularly to the use of ***nearest neighbor text classifiers (NNTCs)*** for filtering [99]. NNTCs are highly effective, but have traditionally been quite inefficient. We implemented, in the Lemur [101] framework, an efficient and effective NNTC implementation using an inverted file, the same technique used in large search engines [58]. The use of an inverted file also opens up the possibility of using a variety of efficiency techniques common in search engines, such as reading inverted lists in order of their impact on scores, quantizing term frequencies to allow compact storage, creating accumulators only for documents with high scores, and using data structures that permit finding the top-scoring documents without sorting all accumulators [22, 84, 92, 97]. We have begun to test these techniques, and are finding that ones that provide an efficiency advantage for query-based text retrieval do not necessarily provide a speedup for NNTC, though work continues on this.

The second approach is random projection into real subspaces of the original huge vector space. There is a theorem [45] which insures that such random projections will - with high probability - preserve the weighted sums, or inner products, which are used in all further calculations. The hope is that randomly projecting document vectors to a much lower dimensional space will increase the efficiency of operations on those vectors. The "fly in the ointment" is that these random projections destroy the sparseness of the document vectors, i.e., the fact that most features (e.g., words, phrases) take on a value of 0 in any given document. The inverted file methods discussed above take advantage of this sparseness, and we have found that the real-valued random projection methods tried so far are more slowed down by the loss of sparseness than they are speeded up by the reduction in dimensionality. However, these methods are very promising for already dense data (e.g., many database records), and may also play a role as an intermediate stage of processing in the Boolean projection methods described next. This work is described in the report [88] and in the papers [58, 59].

The third approach uses random projection into Hamming (Boolean) cubes rather than into a real-valued space. This approach draws on our experiences with the above two approaches, as well as our previous work on discrete randomized methods for both text and other forms of data [25, 55, 26, 56, 77]. Here, as in the work by Boros and Neu [14, 19] described in Section 4 below, the term frequency vectors are converted to Boolean form. The goal here is, as in the second approach, to take advantage of the fact that random projections reduce dimensionality, while (a) maintaining sparsity through as much of the computation as possible, and (b) taking advantage of the Boolean representation to produce highly efficient index structures. The computations here are more elaborate than those for the other methods, but we have a preliminary implementation in place and have good, though not yet competitive, initial results. (See the report [76] and the papers [58, 59].)

The fourth approach seeks to exploit some recent developments in the efficient discovery of "deviant cases" in a stream of vectorized entities. The premise, of course, is that identification of deviants is important in filtering for new events. The description of this work at a very abstract level is complete (see [69] and the papers [58, 59]), which identify some of the implications for filtering). It remains to demonstrate the exact connection to the text filtering problem as defined by our overall goal of monitoring streams of messages. Briefly speaking, we define deviants in the following way. We wish to approximate a data distribution by a model, such as a histogram. We then seek to determine some number of data points to be dropped, with the rest building a histogram with a specific number of bins, so that the overall approximation is "best" in some sense as for instance using an L^2 norm. We build on work of [71], which presents an algorithm that uses sublinear resources on a data stream and determines approximate deviants.

4 Feature Selection and the Batch Filtering Problem

Section 2 discussed the term selection approach that we used in our basic infrastructure-defining task. Our other experiments on term selection, to this point, have been applied in the batch filtering setting. In the batch filtering setting substantial numbers of positive and negative documents are provided at the outset, together with a set of documents representative of the corpus. The task is then to set up a filter once and for all, and apply it to another large collection of documents to select those relevant to each of the included topics. In this setting it is possible to do more complex experiments on the selection of terms to be used for the filtering task. We have considered three methods: (a) combinatorial clustering, (b) a variant using combinatorial principal component analysis, and (c) robust term selection.

4.1 Combinatorial Clustering

In Combinatorial Clustering the sets of terms and documents are considered together, *without reference to the judgments of relevance*, and a subset of terms is chosen in a way that is intended to richly represent the documents [66]. In our experiments the sets of documents considered were variously drawn from the positive judged documents, and from the collection as a whole. The procedure can be visualized in terms of an enormous matrix whose columns are labeled by the documents and whose rows are labeled by the terms. The entries in this matrix are the frequencies with which the terms appear in the documents.

The goal is to find a particularly "rich" subset of terms and documents. Richness is defined for terms by summing the frequencies with which they appear in the selected documents. Richness is defined for documents by summing the number of times the selected terms appear in the document. These row sums and column sums are large when the entries in the sub-matrix are large. The richness of a sub-matrix is defined simply as the minimum of all of these row sums and column sums. In other words, the richness is high when even the weakest of the terms and the weakest of the documents is richly connected to the documents (respectively, terms) remaining in the subset [68, 39, 66, 86, 60].

Using a previously developed computational procedure [68, 39], we are able to find the richest sub-matrix of the enormous term-document matrix, and to place it in a unique chain which contains all such strong sub-matrices. The only adjustable parameter is the size of the sub-matrix retained (particularly the number of terms that it retains). Experiments with this approach, using only the testing material from the TREC batch task, and varying the number of terms retained, have shown that it is possible to achieve substantial accuracy in classifying the texts, while retaining only a few hundred terms. This work was done after the TREC2002 conference, and was not evaluated at TREC. It is described in [6, 7].

4.1.1 Combinatorial PCA

In addition, we have proposed a novel method of dimensionality reduction called combinatorial PCA (principal component analysis). This method was tested on all 100 topics of the TREC-11 Batch Filtering Track data set and we showed that, on average, a reduction of dimensionality from 20,000 to 300 leads to a loss of accuracy between 1% and 3%. Since our experiments do not simulate the exact TREC procedures, we are not yet able to compare these results with the published TREC results. For details of the method and results, see [6, 7].

4.2 Robust Term Selection

Our third approach to term selection makes use of a variety of different measures of "term effectiveness," has been developed in the TREC setting, and was submitted for evaluation in TREC 2002. It scored in the upper half of the batch algorithms reported at this year's competition. It did extremely well on a rather peculiar set of tasks called the "topic intersection tasks." In these tasks, the set of relevant topics was not determined by human evaluation, but was determined by taking the intersection of sets of documents bearing particular labels in the TREC collection. There was considerable discussion at TREC this year about the fact that performance on these artificial topics seemed quite uncorrelated with performance on topics that were being judged by human assessors.

The method starts from the observation that common words (like "the" and "and"), synonyms (or misspellings), etc. greatly inflate the volume of the bag of words required to represent a document. On the other hand, relatively few words are particularly effective at distinguishing the relevant documents from the irrelevant ones. The method employs a heuristic designed to choose a subset of the original set of terms as features. The idea is that this subset will be much smaller than the original one, but still be chosen in such a way that it aids us in distinguishing relevant documents from irrelevant ones. Of course, the method for finding the subset needs to be computationally efficient. Our method uses Boolean bags of words or "bags of bits". For a general survey of computationally efficient approaches to feature selection in the bag of bits context, see [16].

While the details of this term selection method are rather complex, it may be visualized by saying that in it one greedily selects a first term, then a second term, then a third term and so forth. A method for selecting terms is preferred if the discriminating power of the resulting set of terms *tends to increase linearly* with the number of terms included. Somewhat surprisingly, we found that for a number of apparently reasonable methods of term selection this linearity breaks down very quickly and the effectiveness of the set of selected terms grows very slowly. This phenomenon is not well understood. Intuitively one might propose that it represents the selection of terms which are - in some sense - redundant to those terms already selected. While this remains to be understood, the effectiveness of the method itself, described by Boros and Neu [19] and by Boros [14], is established objectively by the TREC results.

The most effective measure of term discriminating power as found by experimentation can be interpreted as a measure that maximizes the "Hamming distance" between the relevant and the not relevant documents. (The ***Hamming distance*** is the number of places on which the Boolean vectors representing the documents differ.) The introduction of the Hamming distance follows from the fact that we are dealing with Boolean vectors, and so it is more appropriate than measures based on the L^1, L^2, or L^∞ norms that are widely used with other representations.

In this preliminary work, the threshold level for converting frequencies to Boolean variables was set at 0 so that we simply count 1 if a term appears and 0 if it doesn't. There is quite a bit of work in various fields involving the problem of analyzing a data set consisting of positive and negative examples presented by Boolean vectors representing this type of threshold. The problem arises in a wide variety of applications of knowledge discovery, data mining, learning, and "logical analysis of data" (see for example [2, 8, 15, 28, 34, 65, 78, 94]). Also relevant is the literature on finding fully-specified Boolean functions from partially-defined ones (see for example [17, 18, 28]). Computational learning theory, which often deals with finding a small subset of the variables explaining input data to a large extent, is also relevant. See for example [13, 20, 29, 61], which deal with learning a target concept depending only upon a small number of the attributes.

The surprising result is not that the Hamming measure well separates the relevant and the not relevant documents, since it was chosen for that purpose. What is surprising is its relative robustness when moving from the training collection to the test collection. Also, as mentioned, it is quite robust as the number of terms increases, a property shared only by a simplified version of itself. See [14, 19] for more details.

5 Learning Sparse Bayesian Classifiers for Text Categorization

Our work on the "learning" component of automated text message processing adopts a Bayesian statistical approach to text filtering, building on very recent work on so-called "sparse Bayesian classifiers." Key references include [32, 36, 89, 90]. The scale of practical problems makes it necessary to develop methods beyond standard statistical classification tools. Early work involved modifications in approaches such as Naive Bayes (a model which presumes that the values of all features are stochastically independent, conditioned only on the relevance of the document) and Rocchio. These don't require much computing power, but better algorithms and more powerful machines have spurred an interest in support vector machines and other methods. Bayesian classifiers make use of external information, such as category descriptions, by assigning prior distributions to the parameters characterizing the filtering problem. These methods are closely related to support vector machines [41, 102] and relevance vector machines [90].

Initially we have applied this approach to the batch learning situation, as a foundation for development of associated online algorithms. At the heart of our approach is an *open-ended search* for a linear functional which operates on the bag of words vectors and which, when combined with a suitable monotonic function, provides a good estimator of the probability that the document at hand is relevant to a topic. If the probability can be accurately estimated, then the decision of whether or not to forward a document for human analysis can be optimized in terms of the known costs of missing valuable documents and of having to review irrelevant ones.

Many standard approaches (e.g., Rocchio) specify precise initial values for the vector of model parameters, usually harnessing the original topic statement or the sum of the vectors representing several relevant documents. In contrast, the Bayesian method requires that we specify all prior knowledge about the parameters in the form of a probability distribution. This distribution in turn yields the probability that documents at or near a particular point in the space of documents are relevant. As data accumulate (in the form of labeled documents), that prior distribution evolves into a so-called posterior distribution by way of an updating procedure called "Bayes rule." The choice of the prior distribution is important as it affects the nature and complexity of the resulting filtering rule.

Since document vectors appear in a space with tens or hundreds of thousands of dimensions, the vector of parameters can be enormous. Sparse classifiers address this challenge by using prior distributions that strongly favor vanishing values for the parameters. The result is that parameters naturally deviate from zero if (and only if) the evidence at hand is emphatic. The resulting optimization problem is computationally difficult and we use a variety of numerical algorithms to address it. Specifically, a variant of an "Expectation Maximization" or EM algorithm makes use of the latent variable representation for probit regression developed in [3]. The goal is to find that distribution of the parameter vector that makes the observed evidence most probable, compared to all other distributions on the parameter vector. We make use of algorithms that avoid the implied process of inverting a matrix of dimensionality equal to the number of terms that are being used to describe the documents. The paper [38] describes the work in detail.

There is, corrspondingly, a sparse Bayesian approach to online learning. The goal here is to sequentially update the posterior distribution of the model parameters based on the arrival of new labeled examples. Our present approach estimates the *mode* of the posterior distribution. Prior work [87, 47] shows that this approach is computationally efficient. While work on the online approach is just beginning, we have taken advantage of a literature on online EM Algorithms [91, 72, 83] and "quasi-Bayes" methods [21, 75, 87]. Later work is expected to involve Monte Carlo algorithms to perform online Bayesian learning, using for example the earlier work of [80]. The paper [46] describes this work.

6 Fusion

The remaining component, fusion of methods, has so far been applied formally to fusion of methods for matching, representation, and learning. In fusion, two or more methods are combined to produce results better than those achieved by either method alone. For early literature see [73, 11, 43].

We have developed a number of computational and visualization tools to facilitate our study of the relationships among various matching algorithms, representations, etc. While the problem can be described as a machine learning problem (explore the space of all possible combinations), in these early stages we feel that it is important to develop and to be guided by some intuition as to how the methods interact with each other and why they work as they do. Thus in the early stages of this part of the effort we are working to put "the human in the loop" to facilitate that exploration. Reports in preparation on these subjects are [4, 50].

We have been able to establish that simple linear methods, when tuned to a *specific* topic or problem ("local fusion"), can produce better results in a substantial number of cases, when judged by standard measures of retrieval performance. This is, in some sense, a tautology. That is, if we represent the score assigned to document d by system S as $s(d|S)$, then the family of scoring rules

$$s(\alpha, d|S_1, S_2) = \alpha s(d|S_1) + (1 - \alpha)s(d|S_2)$$

contains the rules for S_1 and S_2 as endpoints. So a simple search for the best value of α should always produce a rule as good as the better of the two rules being combined. In practice, however, specific heuristics for finding the "optimal" value of α may fail to produce the optimal value, particularly when it lies at one end of the interval $[0, 1]$. In this case the "fused rule" will not be better than both input rules.

It is important to note that this type of "local" fusion is particularly relevant for the filtering situation, in which it is possible to amass fairly large sets (several hundred positives and many hundred negatives) as a spin-off of the analyst's work flow. Thus a rule can be developed for application to a very specific topic or profile.

We also have some positive results for the more difficult case in which the fusion rule is selected on the basis of one sample of topics, and then is applied to another sample [4]. If we think of the former case as "local" data fusion for filtering, then this can be thought of as "global" fusion. Under global fusion one imagines that there is a discoverable relation between the scores that two different systems assign to a document, and the relevance of that document, for any topic whatsoever. Our preliminary results show that such global fusion rules can beat an "oracle" a signficant fraction of the time. The oracle is presumed to simply pick the better system for each new topic. In this preliminary work our fusion procedures were applied to a Rocchio scheme, an alternate scheme that considered the ratio of similarity to positive (or negative) centroids, and one of the schemes based on feature selection guided by the Boolean separability criteria described above. Results are summarized in [4]

7 Streaming Data Analysis

Because of the sheer quantity of the data arising in various intelligence applications or the urgency of producing analyses, it often becomes infeasible to store the data in a central database for future access. That is, it becomes necessary to make computations involving the data, and decisions about the data (such as what to keep), during an initial scan as the data "stream" by. We have begun to explore this aspect of data mining in a "streaming" text messaging environment.

Recently various algorithms for analyzing streaming data have arisen from applications requiring immediate decision making based on current information. Examples are intrusion detection and fault monitoring. Data must be analyzed as it arrives, not off-line after being stored in a central database, because the problems involved are so urgent from a time-to-react point of view. Some other applications require such quick reactions for theoretical (as well as practical) reasons because of the infeasibility of processing and integrating the massive amounts of data generated by a myriad of continuously operating sources. For example, external memory algorithms [1] are motivated by the fact that classical algorithms typically will not scale as soon as data sets do not fit in main memory. At some point, data sets become so large as to preclude most computations that require more than one scan of the data, as they stream by.

Transactional and time-series applications exemplify current streaming data analysis systems. ***Transactional applications*** exploit data recording individual events correlating two or more discrete entities. Examples are phone calls between people and purchases over a credit-card network. One common problem is to maintain behavioral profiles of individual entities (customers, for example). Goals include flagging aberrant transactions, i.e., those not indicated by the models and thus potentially being fraudulent; and detecting "paradigm" shifts in prevailing trends. In these applications as well as others, analysis of data streams also highlights difficult new problems in data visualization. For example, how is time-critical information best displayed? Can automatic response systems be created to deal with common cases?

Time-series applications exploit sequences of observations taken over time. Examples are reports from sensors monitoring network equipment; inventory levels of parts in a warehouse; positions of objects in successive celestial surveys; and records of prices of commodities, stocks, and bonds. Analyzing and mining time-series data presents many new challenges. What are similar time-series data? How can they be clustered, e.g., to isolate seminal events that cause many simultaneous or near-simultaneous disruptions among the observed elements? How can we find interesting trends?

Data streams can be thought of as describing distributions from universes of interest. In the context of text/document streams, the universe could be the set of words that appear in documents that pertain to a particular "topic." The goal is to develop alarm systems that, based on various statistics about the data stream, detect significant changes in the stream. As this project evolves, we intend to exploit recent attention to streaming models of data analysis by the theoretical community as well as recent successes in real-time or near-real-time analysis by practitioners, in application to streaming messages. Of particular interest is a body of work dealing with the analysis of statistical summaries of data streams. Sublinear methods for identifying "representative trends" are studied in the paper [33]. Methods for estimating "rarity" in data streams, as for instance in [30], should also be useful, as should be methods for identifying "deviants" in such streams [44, 71]. Monitoring a stream of items and finding the items that appear most frequently will also be a useful tool. Some promising algorithmic work in this direction can be found in [23, 52, 27, 40].

More details about the relation of this approach to the problem of monitoring message streams are given in [70].

8 A Formal Framework for Monitoring Message Streams

Finally, we have a small ongoing effort aimed at developing some model problems that can be solved in closed form or by easy numerical simulation. The goal here is to gain some understanding of the interrelation among some of the more puzzling aspects of the adaptive learning task. In particular, while it is obvious that we learn something when we receive positive judgments, it is less obvious what we learn when we receive negative judgments. Two approches are being explored: (a) parameterized learning curves, and (b) simplfied discrete models.

8.1 Parameterized Learning Models

As an adaptive filtering process proceeds, we can accumulate a running score of how well the filter is working (for example, the cumulated and current rates of success). One way to summarize this is by a cumulated error curve. Presumably the slope of this curve decreases with experience, and provides an indication of how well the learning is progressing.

One approach is to seek a "phenomenological" parameterization of what has been learned. In this model, there is a single dependent variable - which may be thought of as our average utility u per document submitted, or, equivalently, our overall ability to predict correctly - and a single independent variable, which is the number of documents t that had been submitted for judgment.

Having some understanding of the shape of the cumulated error curve would enable us to answer the question "should the next document be submitted for judgment?" Rather than plotting error curves, we can move directly to the quantity of interest, which is the utility achieved by following a particular path. For example, we might have, for each of several possible patterns of features, a cumulated curve of the "score earned" by submitting documents matching a given pattern. Suppose, as a basis model, that these cumulated utility curves (which do represent a learning phenomenon) have the general form

$$u(t) = v_\infty(1 - e^{-t/T})$$

where v_∞ represents the most that can be earned by submitting a document matching the pattern, while T represents the learning mean life. If submission of a document costs c, and the process is to be scored (without discounting, see next section) over a horizon H, the overall expected utility is

$$Eu(H|v_\infty, T) = -cH + \int_0^H v_\infty(1 - e^{-t/T})$$

.

The issue is to estimate the two parameters v_∞, T as promptly as possible, and either give up completely, or concentrate attention on those feature patterns for which the expected utility Eu is positive and large. Whether to submit documents matching a pattern with low but positive expected utility is a problem whose solution depends on the limitation of analyst resources. Basically, if an analyst is idle, it makes sense to submit a marginally useful document. But, if doing so would cause the analyst to be (probably) unavailable when a more promising document comes along, then it should not be submitted.

Work continues on this problem. See the preliminary report [64].

8.2 Discrete Models

We have developed some extremely simple models of the entire learning which we call "Bin World." In Bin World, documents have a single feature, which is labeled by an integer b, initially $b = 1, 2$, representing a bin. Documents with one value of the feature are relevant with some probability $g < 1$. Documents with the other value are never relevant.

The problem is to discover which of the bins contains the relevant documents. This problem is set up to simulate the TREC setting where, if we do not submit a document for judgment, our present score does not change, and we do not learn whether it is relevant. If we do submit a document for judgment and it is relevant, we gain v units of score. But, if the document submitted is not relevant then we lose one unit of score.

Problems of this type can be mapped into so-called "bandit problems." In these problems, a player is trying to determine which of several slot machines (for example) has the best odds. Of course the player must pay for the experience. The issue is to determine the best strategy. Problems are either framed in terms of a finite horizon with zero discounting (which is accurate for the TREC situation, but not for the real world application), or with a discounting of future gains. An introductory discussion with some literature is given in [35]. The filtering task is somewhat different in that the "player" does not have, at each step, the choice among the several slot machines, but only the choice of whether or not to play the one that has been placed in front of him.

If the "game" has an infinite horizon, of course, we can submit documents with impunity until eventually we discover which of the two features corresponds to the relevant documents.

But in a game with finite horizon, we run the risk that we will get "so far in the hole" that it won't help us to know.

The problem is thus characterized by sequential decisions, which are conditioned on our present knowledge about the chance that each of the two feature values corresponds to the documents that are relevant. The present knowledge can take many different values as experience grows, making it difficult to define clearly an optimum strategy. While such problems are, in principle, amenable to solution by a backwards algorithm, the space of possible situations quickly grows unmanageable.

Thus our work is currently focused on heuristics which might provide a very compact description of the present state of knowledge. The complete present state of knowledge would be the number of successful and unsuccessful submissions that have been made of documents belonging to bin b. As in the parameterized model, heuristics would exploit a simple running count such as the number of bins for which the posterior estimate of the probability that documents in this bin are relevant lies above a particular threshold. This approach is described in a paper [48] that is in preparation.

9 Concluding Remarks

This report summarizes the present state of an ambitious experimental program for the development of new algorithms and perhaps even new theories for monitoring of message streams. We have explored one or more approaches to each of the five basic components of the problem, representation, compression, matching, learning, and fusion. Methods we have explored so far are summarized in Table 1.

Component	Approaches
Compression Representation and Matching	Term Selection, Random Projections Robust Term Selection (TS) Combinatorial Clustering, Combinatorial PCA Bag of Words, Bag of Bits
Matching	Euclidean, Hamming NNTCs, Random Projections
Learning	Rocchio, Bayesian
Fusion efforts	Scope: Local, Global Fusion Rules: Linear Models: Rocchio, Ratio, Robust TS (all Bag of Words) Components: Representation, Matching, Learning

Table 1: Methods being explored for each component of the problem of monitoring message streams.

There is no published precedent for an assault on the filtering problem that jointly explores so many possible lines of attack. While we are finding that the the methods of information retrieval (specifically inverted file methods) that have survived some 40 years of evolution are extremely "fit," we see, nonetheless, that there are opportunities to improve upon a fairly mature and robust technology. These center on two strategies for improvement. The first explores specific components of the filtering process, to see whether, in the presence of the other components, improvement can be achieved. Examples of this are the array of dimension reduction methods: term selection based not only on separating power but also on robustness; term selection based only on the corpus (and not requiring judgments) as in the PCA work; dimension reduction by random projection into Hamming cubes. Each of these shows promise of either improving the accuracy of the filtering, or improving its efficiency, with acceptably small loss in the other key dimension. It is anticipated that no scheme will be best for all situations, which leads naturally to the second strategy.

The second strategy presumes that when multiple methods are available, it should be possible to combine them in some way which permits complementary strengths to improve the overall result. This has been explored already at the level of the matching and representation components, with quite encouraging results, and forms the motivation for our fusion work.

However, many more experiments are needed to extract the full potential, not only alone, but also in combination, of the array of methods that have already been developed or applied in this project.

Our project to date has emphasized retrospective (or "supervised") event identification. In later years, we will also study prospective detection (or "discovery" or "unsupervised learning"), where we must decide whether a "new event" is present in the flow of messages. Our goal would not be to label the new document clusters, but simply to alert users that a new cluster has been detected. This is considerably more difficult than what we have been doing so far and would of course be of enormous value as an intelligence tool. In extending our work to the much harder problem of unsupervised learning, we still plan to concentrate on utilizing new methods for the five components defined above, and combining them. It is because the pure supervised learning task is such a challenge for the classifying power of any reasonable method that we will concentrate on "semi-supervised" learning, learning in which human analysts help to focus on features that will be most indicative of change or anomaly and algorithms assess whether incoming documents do or do not deviate "significantly" on those features. New techniques will be needed to represent the data in a way that makes it possible to highlight significant deviation ("abnormality") through an appropriately defined metric and through the use of new clustering algorithms that take into account the analyst-designated features.

References

[1] Abello, J.M., and Vitter, J.S. (eds.), *External Memory Algorithms*, DIMACS Series, Volume 50, American Mathematical Society, Providence, RI, 1999.

[2] Agrawal, R., Imielinski, T., and Swami, A., "Mining association rules between sets of items in large databases," *International Conference on Management of Data SIGMOD 93*, 1993, 207-216.

[3] Albert, J.H., and Chib, S., "Bayesian analysis of binary and polychotomous response data," *J. of the Amer. Stat. Assoc.*, **88** (1993), 669-679.

[4] Anghelescu, A., Boros, E., Fradkin, D., Lewis, D., Menkov, V., Neu, D., Ng, K-B., and Kantor, P., "Prospective Data Fusion for Batch Filtering," http://www.stat.rutgers.edu/˜madigan/mms/safe/fusionFinalChgs.ps.

[5] Anghelescu, A., Lewis, D.D., Menkov, V., and Kantor, P.B., "Training a Rocchio classifier for adaptive filtering," http://athos.rutgers.edu/˜kantorp/MMS/TREC2002/MMS-TREC.ps.

[6] Anghelescu, A., and Muchnik, I., "An experimental evaluation of a combinatorial clustering model for textual data representation," http://mms-01.rutgers.edu/Documents/CombinatorialClustering/Evaluation.v01.pdf.

[7] Anghelescu, A., and Muchnik, I., "Combinatorial clustering for textual data representation in machine learning models," http://mms-01.rutgers.edu/Documents/CombinatorialClustering/Theoretical.v01.pdf

[8] Angluin, D., "Queries and concept learning," *Machine Learning*, **2** (1988), 319-342.

[9] Baeza-Yates, R., and Ribiero-Nieto, B., *Modern Information Retrieval*, ACM Press, Addison-Wesley, NY, 1999.

[10] Bartell, B., Cottrell, G., and Belew, R., "Learning to retrieve information," in Niklasson, L. and Boden, M. (Eds.), *Current Trends in Connectionism: Proceedings of the Swedish Conference on Connectionism*, LEA: Hillsdale, 1995.

[11] Belkin, N., Kantor, P.B., Fox, E., and Shaw J., "Combining the evidence of multiple query representations for information retrieval," *Information Processing and Management*, **31** (1995), 431-448.

[12] Bins, J., and Draper, B., "Feature selection from huge feature sets," *Eighth IEEE Conference on Computer Vision and Pattern Recognition*, 2001, 159-165.

[13] Blum, A., Hellerstein, L., and Littlestone, N., "Learning in the presence of finitely or infinitely many irrelevant alternatives," *J. of Computer and System Sciences*, **50** (1995), 32-40.

[14] Boros, E., "Notes on term selection," http://www.stat.rutgers.edu/˜madigan/mms/safe/boros11-25-02.pdf.

[15] Boros, E., Hammer, P.L., Ibaraki, T., Kogan, A., Mayoraz, E., and Muchnik, I., "An implementation of logical analysis of data," *IEEE Trans. on Knowledge and Data Engineering*, **12** (2000), 292-306.

[16] Boros, E., Horiyama, T., Ibaraki, T., Makino, K., and Yagiura, M., "Finding essential attributes from binary data," *Annals of Mathematics and Artificial Intelligence*, to appear. (Preliminary version: Technical Report 2000-10, DIMACS Center, Rutgers University, 2000.)

[17] Boros, E., Ibaraki, T., and Makino, K., "Error-free and best-fit extensions of a partially defined Boolean function," *Information and Computation*, **140** (1998), 254-283.

[18] Boros, E., Ibaraki, T., and Makino, K., "Logical analysis of binary data with missing bits," *Artificial Intelligence*, **107** (1999), 219-264.

[19] Boros, E., and Neu, D.J., "Feature selection and batch filtering," http://athos.rutgers.edu/˜kantorp/MMS/BorosTrec2002Report.ps.

[20] Bshouty, N., and Hellerstein, L., "Attribute-efficient learning in query and mistake-bound models," *J. of Comput. Syst. Sci.*, **56** (1998), 310-319.

[21] Chai, K.M.A., Chieu, K.L., and Ng, H.T., "Bayesian online classufiers for text classification and filtering ages," *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 97-104.

[22] Buckley, C., and Lewit, A.F., "Optimization of inverted vector searches," *Eighth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1985, 97-110.

[23] Charikar, M., Chen, K., and Farach-Colton, M., "Finding frequent items on data streams,"*ICALP*, 2002.

[24] Charikar, M., Guruswami, V., Kumar, R., Rajagopalan, S., and Sahai, A., "Combinatorial feature selection problems," *IEEE Symposium on Foundations of Computer Science*, 2000, 631-640.

[25] Cohen, E., and Lewis, D.D., "Approximating matrix multiplication for pattern recognition tasks," *Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1997, 682-691.

[26] Cohen, E., and Lewis, D.D., "Appxoximating matrix multiplication for pattern recognition tasks," *J. of Algorithms*, **30** (1999), 211-252.

[27] Cormode, G., and Muthukrishnan, S., "What is hot and what is not: Dynamically maintaining frequent items," manuscript, Rutgers University Computer Science Department, 2002.

[28] Crama, Y., Hammer, P.L., and Ibaraki, T., "Cause-effect relationships and partially defined Boolean functions," *Annals of Operations Research*, **16** (1988), 299-326.

[29] Damaschke, P., "Adaptive vs. nonadaptive attribute-efficient learning," *Proceedings of the 30th Annual ACM Symposium on the Theory of Computing*, 1998, 590-596.

[30] Datar, M., and Muthukrishnan, S., "Estimating rarity and similarity in data streams," *ESA*, 2002.

[31] Dhillon, I.S., and Modha, D.S., "Concept decompositions for large sparse text data using clustering," *Machine Learning*, **42** (2001), 143-175.

[32] Efron, B., Hastie, T., Johnstone, I. and Tibshirani, R. (2003), "Least angle regression," *Annals of Statistics*, to appear.

[33] Ergun, F., Muthukrishnan, S., and Sahinalp, C., "Sublinear methods for identifying representative trends," manuscript, Rutgers University Computer Science Department, Nov. 2002.

[34] Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R., *Advances in Knowledge Discovery and Data Mining*, AAAI Press/The MIT Press, 1996.

[35] Ferguson, T. S., "Optimal Stopping and Applications," preprint, Mathematics Department, UCLA, http://www.math.ucla.edu/˜tom/Stopping/Contents.html

[36] Figueiredo, M.A.T., "Adaptive sparseness using Jeffreys prior," *Neural Information Processing Systems*, Vancouver, December 2001.

[37] Frakes, W.B., and Baeza-Yates, R. (eds.), *Information Retrieval: Data Structures and Algorithms*, Prentice-Hall PTR, 1992.

[38] Genkin, A., Lewis, D.D., and Madigan, D., "Sparse Bayesian classifers for text categorization - batch learning," http://www.stat.rutgers.edu/˜madigan/mms/safe/batch.ps

[39] Genkin, A., and Muchnik, I.B., "Fixed points approach to clustering," *J. of Classification*, **10** (1993), 219-240.

[40] Gibbons, P., Matias, Y., and Poosala, V., "Fast incremental maintenance of approximate histograms," *VLDB'97*.

[41] Girosi, F., "An equivalence between sparse approximation and support vector machines," *Neural Computation*, **10** (1998), 1445-1480.

[42] Grossman, R., Kantor, P., and Muthukrishnan, S., "CCR-DIMACS Workshop on Data Mining for Homeland Security," http://www.stat.rutgers.edu/˜madigan/mms/CCR-DIMACS.pdf.

[43] Ibraev U., Ng, K.B., and Kantor, P., "Exploration of a geometric model of data fusion," *Proceedings of the 2002 Conference of the American Society for Information Science and Technology*, 2002.

[44] Jagdish, H., Koudas, N., and Muthukrishnan, S., "Mining deviants in time series databases," *Proc. VLDB*, 1999.

[45] Johnson, W.B., and Lindenstrauss, J., "Extensions of Lipschitz mappings into Hilbert space," *Contemporary Mathematics*, **26** (1984), 189-206.

[46] Ju, W-H., and Madigan, D., "Sparse Bayesian classifiers for text categorization - online learning," http://www.stat.rutgers.edu/˜madigan/mms/safe/online.ps

[47] Ju, W-H., Madigan, D., and Scott, S., "On Bayesian learning of sparse classifiers," DIMACS Technical Report 2003-008, DIMACS Center, Rutgers University, 2003, http://www.stat.rutgers.edu/˜madigan/PAPERS/sparse3.pdf.

[48] Kantor, P., "BinWorlds and Bandits: Models for Adaptive Filtering," *in preparation*.

[49] Kantor, P.B., "Infrastructure work," *in preparation*.

[50] Kantor, P.B., "Scoring rules," *in preparation*.

[51] Kantor, P., and Fradkin, D., "Rocchio Model Homotopy Results," *http://dimacspc6.rutgers.edu/˜dfradkin/homotopy/results.html*

[52] Karp, R., Papadimitriou, C., and Shenker, S., "Finding frequent elements in streams and bags," manuscript, 2002.

[53] Kittler, J., Pudil, P, and Somol, P., "Advances in statistical feature selection," in S. Singh, N.A. Murshed, and W.G. Kropatsch (eds.), *Advances in Pattern Recognition – ICAPR 2001 – Second International Conference, Rio de Janeiro, Brazil*, Springer Lecture Notes in Computer Science, **2003** (2001), 425-434.

[54] Kohavi, R., and John, G.H., "Wrappers for feature subset selection," *Artificial Intelligence*, **97** (1997), 273-324.

[55] Kushilevitz, E., Ostrovsky, R., and Rabani, Y., "Efficient search for approximate nearest neighbor in high dimensional spaces," *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, New York, NY, 1998, 614-623.

[56] Kushilevitz, E., Ostrovsky, R., and Rabani, Y., "Efficient search for approximate nearest neighbor in high dimensional spaces," *SIAM J. on Computing*, **30** (2000), 457-474.

[57] Lewis, D., "Feature Selection and Feature Extraction for Text Categorization," *Proceedings of Speech and Natural Language Workshop*, Morgan Kaufmann, San Mateo, California, (1992), 212-217,

[58] Lewis, D.D., and Menkov, V., "A closer look at nearest neighbor text classification," manuscript, 2003.

[59] Lewis, D.D., Muthukrishnan, S., Ostrovsky, R., and Strauss, M., "Towards fast, effective nearest neighbor text classification," http://www.stat.rutgers.edu/~madigan/mms/safe/lewis.ps

[60] Libkin, L.O., Muchnik, I.B., and Shvartser, V., "Quasi-linear monotonic systems," *Automation and Control*, **50** (1989), 1249-1259.

[61] Littlestone, N., "Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm," *Machine Learning*, **2** (1988), 285-318.

[62] Luhnm H.P., "The automatic creation of literature abstracts," *IBM Journal of Research and Development*, **2** (1958), 159-165.

[63] Luhn, H.P., "Keyword-in-context for technical literature (KWIC index)" ASDD Report No. RC-127, IBM Corporation. Yorktown Heights, NY. 1959.

[64] Madigan, D., "Formal framework for MMS," in preparation.

[65] Mannila, H., Toivonen, H., and Verkamo, A.I., "Efficient algorithms for discovering association rules," in U.M. Fayyad and R. Uthurusamy (eds.), *AAAI Workshop on Knowledge Discovery in Database*, 1994, 181-192.

[66] Mirkin, B., and Muchnik, I., "Combinatorial optimization in clustering," in D.Z. Du and P.M. Pardalos (eds.), *Handbook of Combinatorial Optimization*, Kluwer, 1998, 261-329.

[67] Mitchell, T.M., *Machine Learning*, McGraw-Hill, 1997.

[68] Muchnik, I.B., and Shvartser, L.V., "Maximization of generalized characteristics of functions of monotone systems," *Automation and Remote Control*, **52** (1991), 1562-1572.

[69] Muthukrishnan, S., "A simple, coarse vector nearest neighbor algorithm," http://www.stat.rutgers.edu/~madigan/mms/safe/muthu1.ps.

[70] Muthukrishnan, S., "Some statistical analysis algorithms on data streams," http://www.stat.rutgers.edu/~madigan/mms/safe/muthu2.ps.

[71] Muthukrishnan, S., Shah, R., and Vitter, J., "Mining deviants on data streams," manuscript, Rutgers University Computer Science Department, Nov. 2002.

[72] Neal, R.M., and Hinton, G.E., "A new view of the EM algorithm that justifies incremental, sparse, and other variants," in M.I. Jordan (ed.), *Learning in Graphical Models*, Kluwer, 1998, 355-368.

[73] Ng, K-B., *An Investigation of the Conditions for Effective Data Fusion in Information Retrieval*, PhD thesis, Rutgers The State University oF New Jersey - New Brunswick, 1998.

[74] Ogilvie, P., and Callan, J., "Experiments using the Lemur toolkit," in E.M. Voorhees and D.K. Harman (eds.), *The Tenth Text Retrieval Conference TREC 2001*, Gaithersburg, MD, Department of Commerce, National Institute of Standards and Technology, 2001, 103-108.

[75] Opper, M., "A Bayesian approach to online learning," in D. Saad (ed.), *On-line Learning in Neural Networks*, Cambridge University Press, 1998.

[76] Ostrovsky, R., "Nearest neighbor methods based on random projection into Hamming cubes," in preparation.

[77] Ostrovsky, R., and Rabani, Y., "Method for determining approximate Hamming distance and approximate nearest neighbors of a query," United States patent 6.226.640, May 1, 2001.

[78] Quinlan, J.R., "Induction of decision trees," *Machine Learning*, **1** (1986), 81-106.

[79] Qu, Y., Eilerman, A.N., Jin, H., Evans, D.A. "The effects of pseudo-relevance feedback on MT-based CLIR," RIAO 2000, *Content-based Multi-Media Information Access.* CSAIS, Paris, France, 2000, 46-60.

[80] Ridgeway, G., and Madigan, D., "Bayesian analysis of massive dataset via particle filters," *Proceedings of KDD-02, The Eighth International Conference on Knowledge Discovery and Data Mining*, 2002.

[81] Robertson, S.E., and Jones, F.S., "Relevance weighting of search terms," *Journal of the American Society for Information Science*, **27** (1976), 129-146.

[82] Rocchio, Jr., J.J., "Relevance feedback in information retrieval," in Gerard Salton (ed), *The SMART Retrieval System: Experiments in Automatic Document Processing*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, (1971), 313-323.

[83] Sato, M., and Ishii, S., "On-line EM algorithm for the normalized gaussian network," *Neural Computation*, **12** (2000), 407-432.

[84] Scholer, F., Williams, H.E., Yiannis, J., and Zobel, J., "Compression of inverted indexes for fast query evaluation," in M. Beaulieu, R. Baeza-Yeates, S-H. Myaeng, and K. Jarvelin (eds.), *Proceedings of SIGIR2002: The Twenty-fifth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Association for Computing Machinery, New York, NY, 2002, 222-229.

[85] Shaw, J.A., and Fox, E.A., "Combination of multiple searches," *Text REtrieval Conference*, 1994.

[86] Shvartser, L., Kulikowski, C., and Muchnik, I., "Multiple sequence alignment using the quasi-concave function optimization based on the DIALIGN combinatorial structures," Technical Report 2001-02, DIMACS Center, Rutgers University, 2001.

[87] Smith, A.F.M., and Makov, U., "A quasi-Bayes sequential procedure for mixtures," *J. of the Roy. Stat. Soc. (Series B)*, **40** (1978), 106-112.

[88] Strauss, M., "Distances from sketches," http://www.stat.rutgers.edu/~madigan/mms/safe/strauss.ps.

[89] Tibshirani, R., "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society (Series B)*, **58** (1995), 267-288.

[90] Tipping, M.E., "Sparse Bayesian learning and the relevance vector machine," *Journal of Machine Learning Research*, **1** (2001), 211-244.

[91] Titterington, D.M., "Recursive parameter estimation using incomplete data," *J. of the Roy. Stat. Soc. (Series B)*, **46** (1984), 257-267.

[92] Turtle, H., and Flood, J., "Query evaluation: Strategies and optimizations," *Information Processing and Management*, **31** (1995), 831-850.

[93] The Text REtrieval Conference, http://trec.nist.gov

[94] Valiant, L.G., "A theory of the learnable," *Communications of the ACM*, **27** (1984), 1134-1142.

[95] Vogt, C.C., *Adaptive Combination of Evidence for Information Retrieval*, Ph.D. Thesis, University of California, San Diego, 1999.

[96] Vogt, C.C., and Cottrell, G.W., "Fusion via a linear combination of scores," *Information Retrieval*, **1** (1999), 151-173.

[97] Witten, I.H., Moffat, A., and Bell, T.C., *Managing Gigabytes: Compressing and Indexing Documents and Images*, Van Nostrand Reinhold, New York, 1994.

[98] Xu, H., Yang, Z., Wang, B. Liu, B., Cheng, J., Liu, Y., Yang, Z., and Cheng, X., "TREC 11 experiments at CAS-ICT: Filtering and web". http://trec.nist.gov/pubs/trec11/papers/cas_final.hongbo.pdf

[99] Yang, Y., "Expert network: Effective and efficient learning from human decisions in text categorization and retrieval," in W. B. Croft and C.J. van Rijsbergen (eds.), *SIGIR 94: Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, Springer-Verlag, London, 1994, 13-22.

[100] Yang, Y., and Pederson, J.O., "A comparative study on feature selection in text categorization," in D.H. Fisher (ed.), *Proceedings of ICML-97, 14th International Confernce on Machine Learning*, Morgan Kaufmann Publishers, San Francisco, 1997, 412-420.

[101] Zhai, C., Truong, T.N., Lafferty, J., Callan, J., Fisher, D., Feng, F., Allan, J., and Croft, B., *Lemur*, 2001, http://www-2.cs.cmu.edu/˜lemur.

[102] Zhang, T., and Oles, F., "Text categorization based on regularized linear classifiers," *Information Retrieval*, **4** (2001), 5-31.

Acknowledgements: The authors thank the KD-D Group for its support through National Science Foundation grant number EIA-0087022 to Rutgers University. PBK is supported in part by the ARDA AQUAINT program. The authors also thank the members of the DIMACS Monitoring Message Streams Project for detailed comments and input into the preparation of this paper. Thanks go to Andrei Anghelescou, Endre Boros, Dmitry Fradkin, Alex Genkin, Dave Lewis, David Madigan, Vladimir Menkov, Ilya Muchnik, S. Muthukrishnan, Rafail Ostrovsky, and Martin Strauss.