Information Retrieval and Information Theory

Adam Berger
Thesis proposal
aberger@cs.cmu.edu

May 6, 1999

Committee:
John Lafferty (chair)
Jaime Carbonell
Daniel Sleator
Jamie Callan (UMass Amherst and CMU Language Technologies Institute)
Jan Pederson (Infoseek)

Abstract

Information retrieval is concerned with how to classify information and how to judge the similarity between two objects, such as written documents. As the amount of information available in digital form has grown, so too has the need for accurate and scalable algorithms for handling this information. Information theory is concerned with the production and transmission of information. Using a framework known as the source-channel model of communication, information theory has established theoretical bounds on the limits of data compression and communication in the presence of noise and has contributed to practical technologies as varied as cellular communication and statistical translation.

In this work we establish a set of connections between information retrieval and information theory. In particular, we focus on two central and well-studied problems in retrieval: classifying documents and ranking documents by relevance to a query. Viewing these problems from a source-channel perspective, we develop new statistical algorithms for each: to the classification problem we apply methods from error-correcting coding theory, and to the ranking problem we apply methods from statistical translation. In both cases, we report on the architecture and empirical behavior of prototype systems, whose performance improves on that of state of the art techniques.

1 Two Problems in Information Retrieval

For the purposes of this thesis, information retrieval (IR) will denote the analysis and processing of a large heterogeneous collection of text documents. As a scientific field of inquiry, information retrieval predates the digital computer, with roots in bibliometrics, the study of how to organize and characterize written information. However, the dramatic recent increase in availability of information in digital form has transformed the field and, at the same time, introduced new challenges. We are concerned in this work with two core IR problems: document retrieval and classification.

1.1 Document retrieval

Document retrieval is the task of ranking a collection of documents by relevance to a query. Retrieval has received much attention of late, though a large portion of this work is proprietary: commercial Internet search engines such as `Lycos` and `Excite` devote considerable resources to solving the problem of discovering, among the millions of documents comprising the World Wide Web, those web pages most relevant to a user's query. An old and still pressing issue in text retrieval is how to account for the fact that language is nuanced, and words and meanings aren't in a one-to-one relation. Information scientists have given these lexical obstacles a name: *polysemy* refers to the case when a word has multiple meanings, and *synonymy* to the case where two words have the same meaning. A retrieval algorithm which doesn't account for the polysemous nature of LEMON may err in ranking as highly relevant to the query LEMON LAW a document on

FDA citrus-growing regulations; likewise, an algorithm not accounting for synonymy might, for the query AUTOMOBILES, overlook documents containing CAR but not AUTOMOBILE.

The traditional approach to retrieval, called the vector-space method [37], considers two documents to be similar if the word distributions in the two documents are similar. Without some extra embellishments such as query refinement and term expansion [41], this approach does not handle polysemy or synonymy appropriately. The algorithms we propose in Section 4 provide a natural framework for handling the many-to-many relationship between words and meanings.

1.2 Document classification

Document classification is the task of learning, from a collection of categorized documents, how to assign categories to documents. Classifying documents by computer—for instance, automatically assigning index terms to medical research papers [43]—has been of interest to information scientists for many years. In recent years, Internet-related classification research has addressed the problem of learning to collect interesting postings to electronic discussion groups based on a user's predilections [25], automatically classifying web pages by content [15], and suggesting web pages to a user based on his or her expressed preferences [29].

The work we describe in Section 3 focuses on a restricted version of the general classification problem, where documents have exactly one correct labeling. In this case the map from documents to categories is actually a function. The text databases we employ for experimental purposes all contain singly-labeled documents; in addition, each category appears very often in the data. Under these conditions, the statistical technique known as Naive Bayes classification is highly effective. Nonetheless, Section 3.3 demonstrates that in this setting, a technique developed in the artificial intelligence and statistics literature called *error-correcting output coding* (ECOC) considerably outperforms Naive Bayes. Further experiments reported there suggest that ECOC will also perform well in sparse-data domains. One of the goals of this thesis work—an initial foray appears in Section 3.1—is to try to understand why ECOC works so well in general and for document classification in particular. Another goal is to extend the ECOC framework to handle more general text collections: those with sparsely-represented labels and multiply-labeled documents.

2 The source-channel model

Figure 1: The source-channel model in information theory

The field of information theory, as old as the digital computer, concerns itself with the efficient, reliable transmission of information. Figure 2 depicts the standard information theoretic view of communication. In some sense, information theory is the study of what belongs in the boxes in this diagram.

Encode: Before transmitting some information s across an unreliable channel, the sender can add redundancy to it, so that noise introduced in transit can be identified and corrected by the receiver. This is known as *error-correcting coding*. We represent encoding by a function $\psi : s \to x$.

Channel: There are different ways to model how information is compromised in traveling through a channel, but typically one characterizes a channel's behavior by a conditional probability distribution $p(y \mid x)$, where x is a random variable representing the input to the channel, and y the output.

Decode: The inverse of encoding: given a message s which was encoded into $\psi(s)$ and then corrupted via $p(y \mid \psi(s))$, recover the original message. Assuming the source emits messages s according to some known distribution[1] $p(s)$, decoding amounts to finding

$$s^\star = \operatorname*{argmax}_s p(\psi(s) \mid y)$$

[1] This document intentionally sacrifices notational rigor for clarity. In particular, p will stand for several different conditional and unconditional probability distributions. Furthermore, the parameter(s) of a distribution p will stand for either random variables or the values taken by these variables. The proper meaning should always be clear from context.

$$= \operatorname*{argmax}_s p(y \mid \psi(s))\, p(s), \tag{1}$$

where the second equality follows from Bayes' Law.

The classical application of information theory is communication between source and receiver separated by some distance. Deep-space probes and cellular phones, for example, use a form of codes based on polynomial arithmetic in a finite field to guard against losses and errors in transmission. Error-correcting codes are also becoming popular for guarding against packet loss in Internet traffic, where the technique is known as *forward error correction* [19].

The source-channel framework has also found application in settings seemingly unrelated to communication. For instance, the now-standard approach to automatic speech recognition views the problem of transcribing a human utterance from a source-channel perspective [2]. In this case, the source message is a sequence of (English, say) words s. In contrast to communication via error-correcting codes, we aren't free to select the code here—rather, it's the product of thousands of years of linguistic evolution. The encoding function maps a sequence of words to a pronounciation x, and the channel "corrupts" this into an acoustic signal y. The decoder's responsibility is to recover the original word sequence s, given

- the received acoustic signal y,
- a model $p(y \mid x)$ of how words sound when voiced,
- a knowledge of the encoding function, and
- a model $p(x)$ of what sorts of word sequences are likely and unlikely.

One can also apply the source-channel model to language translation. Imagine that the person generating the text to be translated originally thought of a string x of English words, but the words were "corrupted" into a French sequence y in writing them down. Here the channel is purely conceptual, but no matter; decoding is still a well-defined problem of recovering the original English x, given the observed French sequence y, a model $p(y \mid x)$ for how English translates to French, and a prior $p(x)$ on English word sequences [6].

The lessons of statistical methods in speech recognition, natural language processing, and machine translation suggest the potential benefit of applying a source-channel paradigm to problems in information retrieval. Far more than a simple application of Bayes' law, there are compelling reasons why the ritual of turning a search problem around to predict the input should be rewarding. When designing a statistical model for language processing tasks, often the most natural route is to build a *generative* model which builds the output step-by-step. Yet to be effective, such models need to liberally distribute probability mass over a huge space of possible outcomes. This probability can be difficult to control, making an accurate *direct* model of the distribution of interest difficult to fashion. Time and again, researchers have found that predicting what is already known from competing hypotheses is easier than directly predicting all of the hypotheses.

2.1 Preview of this document

The rest of this document will describe in detail how the source-channel model applies to classification and retrieval, but we pause here to sketch how these two problems fit into this framework.

For document retrieval, we take the view that in formulating a query, a user is distilling an information need into a succinct representation. The goal of a retrieval algorithm is to discover, among the documents available to it, which are closest to this information need. The distillation plays the role of a noisy channel, corrupting the information need into a query; retrieval is a decoding step, from query to back to document(s).

For single-label document classification, we take the source message—the message to be recovered—to be the correct label of the document. The encoding step converts this label into a member of an error-correcting code. The channel will corrupt this codeword by changing some of its bits. These corrupted bits will correspond (in a way made clear in Section 3) to errant binary classifiers. As with the source-channel model for machine translation, the encoding and channel are purely conceptual: the classification step occurs in decoding, when the correct label is (hopefully) recovered by comparing the received bitvector to all codewords, and taking the label corresponding to the closest codeword.

Table 2.1 shows how the source-channel framework applies to the aforementioned collection of problems. We emphasize that the source-channel framework is only a restatement of a problem, not a solution: the goal of this thesis is not to frame a problem in a new way, but to show that doing so leads to practical and high-performance algorithms. Details of the first three columns are left to the references [27] [23] [6]. The last two rows comprise the thesis work proposed here, and are also the focus of the StART project (Statistical methods Applied to Retrieval Technology) at Carnegie Mellon University. We take up these topics in the following sections.

Applying the source-channel model

	source	*encoder*	*channel*	*decoder*
point to point digital communication	packets	add redundancy with forward error-correction (FEC)	e.g., packets dropped independently with a constant probability	use encoded redundancy to recover source
automatic speech recognition	English word sequence, following distribution p(E)	map words to pronounciation G	corrupted to (voiced as) an acoustic signal A according to p(A | G)	recover English words from acoustic signal A, channel model p(A | G), encoding function, and p(E)
machine translation	(same)		corrupted (translated) to French words F according to p(F | E)	recover English words from French F and models p(F | E) and p(E)
document retrieval	information need I	from I, generate an "ideal document" D	corrupted (distilled) to a query Q according to p(Q | D)	find documents D′ in collection which maximize p(D′ | Q)
document classification	label of document	convert label to a bitvector (codeword)	some bits flipped	output label of codeword nearest to received bitvector

3 Classifying documents with error-correcting codes

This section discusses the application of error-correcting output coding (ECOC) to the task of document categorization. ECOC, of recent vintage in the AI literature, is a method for decomposing a multiway classification problem into many binary classification tasks, and then combining the results of the subtasks into a hypothesized solution to the original problem. There has been much recent interest in the machine learning community about algorithms which integrate "advice" from many subordinate predictors into a single classifier, and error-correcting output coding is one such technique. We provide experimental results on several real-world datasets, extracted from the Internet, which demonstrate that ECOC can offer significant improvements in accuracy over conventional classification algorithms.

Error-correcting output coding works in two stages: first, independently construct many subordinate classifiers, each responsible for removing some uncertainty about the correct class of the input; second, apply a voting scheme to decide upon the correct class, given the output of each weak learner. Recent experimental work has shown that ECOC offers improvements over standard k-way classification methods in domains ranging from cloud classification [1] to speech synthesis [4], and a number of theories have been proposed for its success [21]. Here we explore the application of error-correcting output coding to document categorization.

The idea of "classifying by consensus" using a large number of independently-constructed classifiers has appeared in a number of other guises recently in the machine learning literature. The technique of bagging, for instance, involves generating multiple training sets by sampling with replacement, learning a classifier from each generated set, and allowing the learned classifiers to vote on the correct class for a

unlabeled object [8]. Boosting can be viewed as a special case of bagging where the sampling is adaptive, concentrating on misclassified training instances [18]. Voting methods have also been applied to combining multiple neural networks trained on the same data [30] and applying different types of classifiers to the same problem [33].

Why consensus algorithms work so well in practice is still an open question. As a step in that direction, theoretical work has recently established that combining multiple runs of a classification algorithm can reduce its variance [9]. Unlike most voting algorithms, the constituent classifiers in error-correcting output coding aren't all solving the same problem; in fact, they are each solving a distinct binary classification problem. It has been shown [24] that this property of the ECOC algorithm bestows on it, in addition to the variance-reduction property of all voting methods, the ability to correct for bias in the constituent classifiers. Section 3.2 suggests some reasons, from a statistical and combinatorial viewpoint, why the ECOC method should work as well as it does, but there is much more work to be done in this area.

We are here interested in applying ECOC to the problem of text categorization: given a database of documents, each annotated with a single label or set of labels, learn a function $\Lambda : x \rightarrow y$ from document to label. The recent explosion in availability of online text lends an extra importance, if not urgency, to the problem of text classification, and also suggests a source of experimental data. In fact, the experiments reported in Section 3.3 are all conducted on data gathered from the Internet.

These databases have the convenient characteristic that each label is well represented. Under these conditions, the method of Naive Bayes classification is highly effective. However, Section 3.3 demonstrates that in this setting, error-correcting output coding consistently outperforms Naive Bayes. Further experiments reported there suggest that ECOC will be of utility in the sparse-data domain as well, a problem we plan to address in this thesis.

3.1 Error-correcting output coding

We describe the technique of error-correcting output coding with a simple example: the task of classifying newswire articles into the $m = 4$ categories {*politics*, *sports*, *business*, *arts*}. To begin, one assigns a unique n-bit vector to each label (where $n > \log_2 m$):

label	coding
politics	0110110001
sports	0001111100
business	1010101101
arts	1000011010

One can view the ith bitvector as a unique coding for label i. For this reason (and others, which will soon become apparent), we'll refer to the set of bitvectors as a *code* and denote it by $\mathcal{C}$. The ith row of $\mathcal{C}$ we will write as $\mathcal{C}_i$, and the value of the jth bit in this row as $\mathcal{C}_{ij}$.

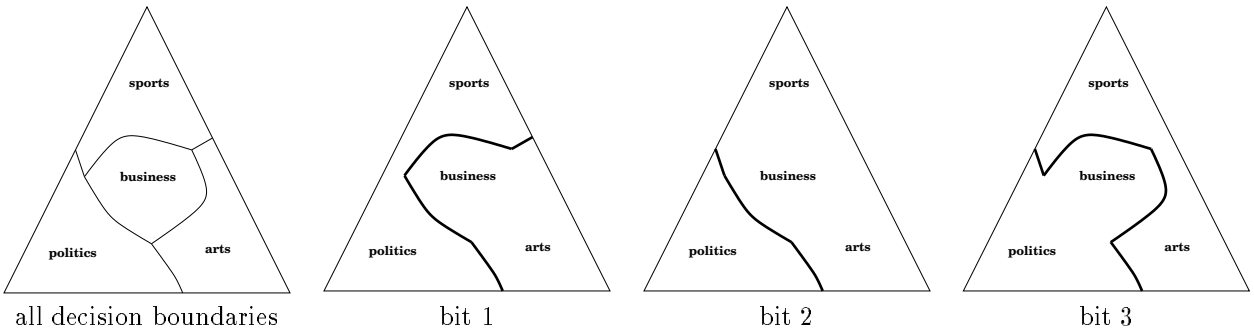

Figure 2: Decision boundaries for the first three plug-in classifiers corresponding to the code given above.

The second step in constructing an ECOC classifier is to build an individual binary classifier for each column of the code—10 classifiers in all, in this case. The positive instances for classifier j are documents

with a label i for which $\mathcal{C}_{ij} = 1$. The third classifier, for instance, has the responsibility of distinguishing between documents whose label is *sports* or *arts* and those whose label is *politics* or *business*. Heeding to convention, we refer generically to any algorithm for predicting the value of a single bit as a "plug-in classifier" (PiC). A PiC, then, is a predictor of whether a document belongs to some fixed subset of the classes.

Algorithm 1
Training an ECOC document classifier

Input: Documents $\{x_1, x_2, \ldots x_D\}$;
Labelings $\{y_1, y_2, \ldots y_D\}$ (with m distinct labels);
Desired code size $n \geq \log_2 m$

Output: m by n coding matrix $\mathcal{C}$;
n classifiers $\{\lambda^1, \lambda^2 \ldots \lambda^n\}$

1. Generate a m by n 0/1 coding matrix $\mathcal{C}$
2. Do for $j \in [1, 2 \ldots n]$
   - Construct two superclasses, $\mathcal{S}_j^+$ and $\mathcal{S}_j^-$. $\mathcal{S}_j^+$ consists of all labels i for which $\mathcal{C}_{ij} = 1$, and $\mathcal{S}_j^-$ is the complement set.
   - Construct a binary classifier λ^j to distinguish $\mathcal{S}_j^+$ from $\mathcal{S}_j^-$.

To summarize, training an ECOC classifier consists of learning a set $\Lambda = \{\lambda^1, \lambda^2 \ldots \lambda^n\}$ of independent binary classifiers. With Λ in hand, one can hypothesize the correct class of an unlabeled document x as follows. Evaluate each independent classifier on x, generating a n-bit vector $\Lambda(x) = \{\lambda^1(x), \lambda^2(x), \ldots \lambda^n(x)\}$. Most likely, the generated bitvector $\Lambda(x)$ will not be a row of $\mathcal{C}$, but it will certainly be closer (in Hamming distance Δ, say) to some rows than to others. Categorizing the document x involves selecting $\text{argmin}_i\, \Delta(\mathcal{C}_i, \Lambda(x))$, the label i for which $\mathcal{C}_i$ is closest to $\Lambda(x)$. (If more than one row of $\mathcal{C}$ are equidistant to $\Lambda(x)$, select one arbitrarily.) For instance, if the generated bitvector $\Lambda(x) = \{1010111101\}$, the document would receive the label *business*.

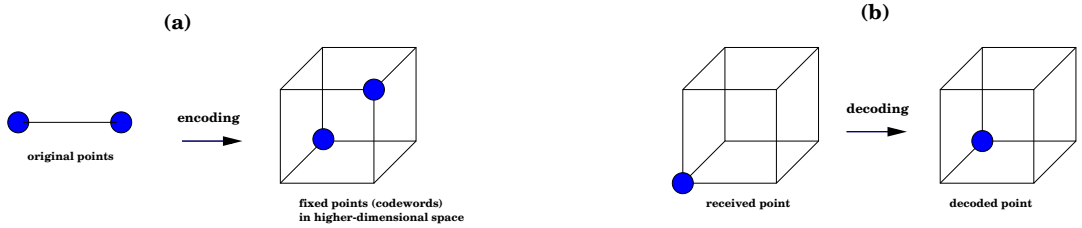

Figure 3: (a) Encoding the two points in $\{0, 1\}$ by mapping them to two well-separated fixed points in $\{0, 1\}^3$. (b) Decoding involves selecting the closest fixed point to the point received. This simple code is robust to a single corruption between the encoding and decoding steps.

To the extent that rows of $\mathcal{C}$ are well-spaced in Hamming distance, the classifier will be robust to a few errant PiCs. This is exactly the idea behind error-correcting codes as well: to transmit a point in the m-dimensional cube reliably over a noisy channel, map it to one of a set of well-separated "fixed points" in a higher-dimensional cube; to recover the original point, find the closest fixed point to the point actually received and take its preimage in the original cube (See Figure 3).

In general, $\lambda^j(x)$ may not be a $0/1$ value, but a real-valued probability, measuring the classifier's confidence that document x belongs in the j'th superclass. In this case, one can search for the nearest neighbor according to some L_p distance, rather than Hamming distance. In the experiments reported in Section 3.3, the plug-in classifiers output a probability, and we compute the nearest neighbor according to L_1 distance. As an aside, we remark that using the "confidence" of individual predictors in a voting scheme appears to be gaining popularity in the machine learning literature [38].

Algorithm 2
Applying an ECOC document classifier

Input: Trained ECOC classifier: m by n coding matrix $\mathcal{C}$ and n classifiers $\{\lambda^1, \lambda^2 \ldots \lambda^n\}$;
Unlabeled document x

Output: Hypothesized label y for x

1. Do for $j \in [1, 2 \ldots n]$
   - Compute $\lambda^j(x)$---the confidence with which PiC j believes $x \in \mathcal{S}_j^+$.
2. Calculate $\Delta(\Lambda(x), \mathcal{C}_i) = \sum_{j=1}^{n} \mid \lambda^j(x) - \mathcal{C}_{ij} \mid$ for $i \in [1, 2 \ldots m]$.
3. Output $\text{argmin}_i \, \Delta(\Lambda(x), \mathcal{C}_i)$

3.1.1 The Naive Bayes classifier

The PiC we relied most heavily on in constructing ECOC classifiers is the *Naive Bayes* classifier [26]. Naive Bayes assumes that a document is generated by selecting a label y according to a prior distribution $p(y)$, and then independently selecting words w for the document according to a distribution $p(w \mid y)$. The probability of generating a document $W = \{w_1, w_2 \ldots w_N\}$ of N words from label y is thus

$$p(W \mid y) = \prod_{i=1}^{N} p(w_i \mid y) \tag{2}$$

Used for prediction, the Naive Bayes classifier selects for an unlabeled document W the most likely label, given by

$$\begin{aligned} \underset{y}{\text{argmax}} \; p(y \mid W) &= \underset{y}{\text{argmax}} p(y) p(W \mid y) \\ &= \underset{y}{\text{argmax}} \; p(y) \prod_{i=1}^{N} p(w_i \mid y) \end{aligned} \tag{3}$$

where the first equality follows from Bayes' Law.

3.1.2 Why should ECOC classification work?

Some standard classification algorithms such as backpropagation [36] are best suited to distinguishing between two outcomes. A natural way to combine such algorithms to predict from among $k > 2$ outcomes is to construct k independent predictors, assigning predictor i the task of deciding whether the ith outcome obtains. To build the classifier, construct m individual classifiers, where the positive examples for classifier λ^i are those documents with label i. To apply the classifier to an unlabeled document x, select $i^\star = \text{argmax}_i \, \lambda^i(x)$—the label whose classifier produces the highest score. This is what some call the *one versus rest* strategy. This method is a special case of ECOC classification where $\mathcal{C}$ is the m by m identity matrix.

To see why one might expect ECOC classification to outperform a one-vs.-rest approach, consider the problem of learning to classifying fruit. Imagine that within the labeled set of examples used to train the individual one-vs.-rest classifiers, the only yellow fruit are bananas. So λ^{banana} will learn a strong association between a yellow color and bananas. Now provide a yellow grapefruit to the trained one-vs.-rest classifier. The value of $\lambda^{\text{grapefruit}}(x)$ will likely be close to one—after all, the object in question is round and grapefruit-sized, despite not being red like all the grapefruits encountered in training. But the value of $\lambda^{\text{banana}}(x)$ will be *very* close to one, and the system will misclassify the object as a banana. ECOC classification is less "brittle" than the one-vs.-rest approach: the distributed output representation means one errant subordinate classifier won't necessarily result in a misclassification. This is a circuitous way of saying that ECOC reduces variance of the individual classifiers.

Many classification algorithms, including decision trees, exponential models, and neural networks have the capability to directly perform multiway ($k > 2$) classification. A reasonable classification strategy with these algorithms is to construct a single, monolithic classifier. But the monolithic classifier faces a difficult

task. Imagining the classes as clouds in a large-dimensional feature space, a single classifier must learn all the decision boundaries simultaneously, whereas each PiC of an ECOC classifier learns only a relatively small number of decision boundaries at once. Moreover, (assuming n is sufficiently large) an ECOC classifier learns each boundary many times, and is forgiving if a few PiCs place the input x on the wrong side of some decision boundaries [24].

3.2 Choosing a good code

Empirical work has established [22] that ECOC classification performs well when the coding matrix $\mathcal{C}$ is constructed randomly—specifically, by choosing each entry $\mathcal{C}_{ij}$ uniformly at random from $\{0, 1\}$. The next two sections provides some statistical and combinatorial arguments for why this should be so.

3.2.1 A statistical perspective

Definition: *Given a database $\mathcal{D}$ of (document, label) pairs (x, y_x) with empirical distribution $\tilde{p}(y, x)$, the* **Bayes classifier** *is $\beta(x) \equiv \text{argmax}_i \, \tilde{p}(y_x = i \mid x)$.*

The Bayes classifier assigns to a document x the label which appears most often in the database $\mathcal{D}$ with x. In terms of classification accuracy on $\mathcal{D}$, the Bayes classifier is the best possible strategy. In the present setting, it is reasonable to assume documents don't occur multiple times with different labels in the collection, and so the Bayes classifier simply selects the label of the document in $\mathcal{D}$. During the training phase, all document labels are available and so we have access to the Bayes classifier. But in applying the classifier we do not. Yet the Bayes classifier will still turn out to be a useful concept, as the following definition and theorem from James [21] suggest.

Definition: *A classification algorithm Λ built from subordinate classifiers $\{\lambda^1, \lambda^2, \ldots\}$ is* **Bayes consistent** *if, whenever the λ^i are Bayes classifiers, so too is Λ.*

Loosely speaking, a Bayes consistent classifier constructed from accurate PiCs will be accurate. This is a property one would like to achieve in an ECOC classifier. The next theorem states conditions under which this is achievable.

Theorem 1 *Assuming $\mathcal{C}$ was constructed randomly, the ECOC classifier becomes consistent as $n \to \infty$.*

This theorem is *not* saying that with enough bits, an ECOC classifier will do arbitrarily well. Consistency of an ECOC classifier doesn't guarantee correctness—since the PiCs aren't themselves producing Bayes estimates. Still, this theorem suggests why a random construction of $\mathcal{C}$ performs well.

3.2.2 A combinatorial perspective

The example code presented earlier has the unfortunate property that the third and tenth columns are equal. Therefore, the corresponding classifiers will learn precisely the same task. This is a permissible situation, though hardly desirable. Not permissible is when two *rows* of $\mathcal{C}$ are equal, for then the code cannot distinguish between the corresponding labels. Fortunately, for a randomly-generated binary code with sufficiently many columns, the probability of such an event is miniscule: for a code with m labels and n bits, the probability is

$$1 - \prod_{i=1}^{m-1} \left(1 - \frac{i}{2^n}\right),$$

which is one for $n = \log_2 m$ but approaches zero quickly thereafter as n increases.

More generally, if two rows in $\mathcal{C}$ are close in Hamming distance, an ECOC classifier built from $\mathcal{C}$ is apt to confuse the corresponding labels. We'll write $\Delta(\mathcal{C}_i, \mathcal{C}_{i'})$ as the Hamming distance between rows i and i' of $\mathcal{C}$, and $\Delta_{\min}(\mathcal{C})$ as the minimum distance between any two codewords. If the PiCs produce binary outputs, then the ECOC classifier can always recover from at least $\lfloor \Delta_{\min}(\mathcal{C})/2 \rfloor$ incorrect PiC outputs.

Early work on error-correcting output coding looked to algebraic coding theory, and in particular to the family of linear codes, for a coding matrix $\mathcal{C}$. An n-bit *linear error-correcting code*, a subspace of the vertices on

a n-dimensional cube, can be defined as the span of an n-column binary matrix $\mathcal{G}$, called a *generator matrix*. Error-correcting codes are often measured on the minimum distance between any two linear combinations of $\mathcal{G}$. BCH codes [27], a popular class of linear algebraic error-correcting codes, have the useful property that their codewords (all different linear combinations of rows of $\mathcal{G}$) are well separated. Using such a matrix for ECOC classification might for this reason seem attractive, and early work on ECOC classification used for $\mathcal{C}$ the generator matrix of a BCH code. But the codes used for ECOC are not linear—the valid codewords are *rows* of a binary matrix, rather than linear combinations thereof. It thus appears that the well-separatedness of the codewords of linear codes is not directly relevant to finding good ECOC codes. And in fact, subsequent ECOC work established that random matrices outperform BCH-type codes. The following theorems provide some justification, from a combinatorial viewpoint, for this observation.

Theorem 2 *For any m by n binary matrix $\mathcal{C}$, there exist two rows which differ in at most $\frac{n}{2}\left(\frac{m}{m-1}\right)$ bits.*

Proof Let H be the minimum distance between any two rows of one such matrix $\mathcal{C}$. Select two rows $i, j \in [1, 2 \ldots m]$ with replacement. Select a column $k \in [1, 2 \ldots n]$. The probability that $\mathcal{C}_{ik} \neq \mathcal{C}_{jk}$ is

$$p_{\text{diff}} \geq \left(\frac{m-1}{m}\right)\frac{H}{n}$$

Now select a column $k \in [1, 2 \ldots n]$, and then select two rows $i, j \in [1, 2 \ldots m]$ with replacement. The probability that $\mathcal{C}_{ik} \neq \mathcal{C}_{jk}$ is no greater than $1/2$. Combining these inequalities to solve for H gives the result. □

This shows that, as m becomes large, a relative spacing of one half is optimal. If we consider only square matrices, there exists an explicit construction which achieves this bound; namely, the Hadamard matrix. For general $0/1$ matrices we are not aware of an explicit construction meeting this bound, but the following result suggests that a random construction is likely to have good separation.

Theorem 3 *Define a well row-separated m by n binary matrix as one in which all rows have a minimum relative Hamming separation at least*

$$\frac{1}{2} - 4\sqrt{\frac{\log m}{n}}$$

The probability that a randomly-constructed binary matrix is not well row-separated is at most $1/m^4$.

Proof Given is a binary matrix $\mathcal{C}$. Fix two different rows r_1 and r_2. For $i \in [1, 2 \ldots n]$, define the random variable x_i as

$$x_i = \begin{cases} +1 & \text{if } r_1[i] = r_2[i] \\ -1 & \text{otherwise} \end{cases}$$

Let $S = \sum x_i$. For a randomly-constructed $\mathcal{C}$ with n even, $E[S] = 0$, which corresponds to an $n/2$ Hamming distance between the rows. We are interested in the probability that $S \gg 0$. Using Chernoff bounds,

$$p(S > 4\sqrt{n \log m}) < e^{-8 \log m} = \frac{1}{m^8}.$$

There are $\binom{m}{2}$ rows in $\mathcal{C}$, and so the probability that no pair of rows is too close is

$$p \leq \binom{m}{2}\frac{1}{m^8} \leq \frac{1}{m^4}$$

□

Although attention in the ECOC literature has generally concentrated on finding a $\mathcal{C}$ with good row separation, a perhaps equally important desideratum is a large separation between columns. Columns that are close give rise to classifiers which are performing nearly the same task—in the extreme case, two equal columns corresponding to two identical classifiers. With a simple change of variables, Theorem 3 shows that random matrices are likely to have good *column* separation as well, providing another justification for constructing a code randomly.

collection	documents	labels	words
20 newsgroups	19997	20	60915
4 universities	8263	7	29004
Yahoo science	10158	41	69939
Yahoo health	5625	36	48110

Table 1: Particulars on the four training datasets used. Each dataset was partitioned five separate times into a $3/4 - 1/4$ training/test split, and the numbers are statistics from the last of these trials. The last column reports the number of distinct words in the collection, excluding those appearing once or twice.

3.3 Experimental results

We applied error-correcting output coding classification to four real-world text collections, all extracted from the Internet. All corpora were subject to the same preprocessing: remove punctuation, convert dates and monetary amounts and numbers to canonical forms, map all words to uppercase, and remove words occurring twice or less. Table 1 summarizes some salient characteristics of these datasets.

- **20 Newsgroups**: This is a collection of about $20,000$ documents, culled from postings to 20 Usenet discussion groups [25]. The documents are approximately evenly distributed among the 20 labels.
- **Four universities**: This (misnamed) dataset contains web pages gathered from a large number of university computer science departments [15]. The pages were manually classified into the categories {*course*, *department*, *faculty*, *staff*, *student*, *project*, *other*}.
- **Yahoo science**: Following Baker *et al.* [3], we automatically extracted the entire *Yahoo* science hierarchy in early 1999, and formed a labeled collection containing 41 classes by collapsing the hierarchy to the first level.
- **Yahoo health**: This corpus was collected in the same way as the science collection, but has rather different characteristics. In particular, many of its 36 classes are highly confusable, presenting a difficult task for classification algorithms. For instance, three of the labels in this collection are *Health Administration*, *Hospitals And Medical Centers*, and *Health Care*.

Figure 4 plots ECOC classification accuracy against code size n for these four corpora. The codes $\mathcal{C}$ were constructed by selecting entries uniformly at random from $\{0, 1\}$, except in the case of the 4 universities dataset, for which the columns of $\mathcal{C}$ were a random permutation of the 126 unique, non-trivial 7-bit vectors.

From an implementation standpoint, a larger value of n incurs a penalty in speed. (This may be an issue in high-throughput systems such as a text filtering systems designed to route relevant news articles to many users, each with their own preferences. However, Figure 4 suggests that, to a point, larger values of n offer more accurate classification. And beyond that point, accuracy doesn't tail off, as is the case in many other learning algorithms for classification, which are prone to overfitting when the number of parameters becomes large.

The four universities dataset was the only collection on which ECOC classification didn't outperform Naive Bayes one-vs.-rest classification. The ECOC classifier's performance on this collection is almost poignant: error rate is steadily decreasing until $n = 126$, at which point there simply are no more unused, non-trivial 7-bit columns to add to $\mathcal{C}$.[2]

In the collections we are considering, each label is well-represented in the data and models $p(w \mid y)$ can be well estimated. In this setting the standard Naive Bayes method is highly competitive [26]. For this reason, we use a Naive Bayes classifier as the PiC in the ECOC classifiers corresponding to Figure 4. However, on datasets with poorly-represented labels, Naive Bayes can starve for a lack of data. With an eye towards such collections, we explored using a feature-based classification approach as the ECOC PiC. Specifically,

[2] It has been pointed out that by splitting classes into subclasses—by random partitioning, for example, or using an EM-type approach—one could create even longer codes, and in this way hope to reduce the error rate further.

4 **universities**	20 **newsgroups**	*Yahoo* **health**	*Yahoo* **science**
course	*alt.atheism*	*Alternative Medicine*	*Acoustics*
department	*comp.graphics*	*Chat*	*Agriculture*
faculty	*comp.os.ms-windows.misc*	*Children's Health*	*Alternative*
other	*comp.sys.ibm.pc.hardware*	*Conferences*	*Amateur Science*
project	*comp.sys.mac.hardware*	*Diseases and Conditions*	*Ask an Expert*
staff	*comp.windows.x*	*Education*	*Astronomy*
student	*misc.forsale*	*Emergency Services*	*Aviation and Aeronautics*
	rec.autos	*Employment*	*Bibliographies*
	rec.motorcycles	*Environmental Health*	*Biology*
	rec.sport.baseball	*First Aid*	*Chemistry*
	rec.sport.hockey	*Fitness*	*Cognitive Science*
	sci.crypt	*General Health*	*Complex Systems*
	sci.electronics	*Health Administration*	*Computer Science*
	sci.med	*Health Care*	*Dictionaries*
	sci.space	*Health Sciences*	*Earth Sciences*
	soc.religion.christian	*Hospitals and Medical Centers*	*Ecology*
	talk.politics.guns	*Institutes*	*Education*
	talk.politics.mideast	*Long Term Care*	*Employment*
	talk.politics.misc	*Medicine*	*Energy*
	talk.religion.misc	*Men's Health*	*Engineering*

Table 2: Categories for the four classification datasets. For the *Yahoo* datasets, only the first 20 categories are shown.

we trained binary decision trees to predict the individual bits in an ECOC code; the questions at the nodes of each tree were of the form *Did word w appear in the document?* We do not expect such a classifier to match the best reported performance on this dataset; McCallum $et\ al.$ [28] provide evidence that classifiers which consider only *whether* a word occurs in a document and not how often it occurs tend to perform worse than classifiers with access to word count information. However, Figure 5 does show that for sufficiently high n, combining decision trees into an ECOC classifier improves performance over a one-vs.-rest decision tree approach, which augurs well for the application of ECOC to larger, sparse datasets.

Although up to a point accuracy tends to monotonically improve with n, this isn't always the case. Figure 6 plots accuracy against code size for the four universities collection, using a different partitioning method than in Figure 4. (Specifically, we set aside for testing not a random partition of the data, but those 1197 web pages from the University of Washington. This has the effect of making the problem more challenging, since automatic classification techniques cannot exploit intra-university idiosyncrasies.) The most notable feature of the plot is the spike at $n = 4$, and again at $n = 8$. Within this range, the error rate is considerably below the one-vs.-rest result. Such behavior seems implausible. After all, a four bit code cannot accommodate seven entries without placing two codewords adjacent to each other in Hamming space, which leaves no room for error.

Table 2 shows the first eight bits of this code. Notice that the first four bits don't distinguish among *other*, *project*, *staff*, and *student*. Figure 7a show that among these classes, *other* occurred most often in training, and thus has the largest prior. So a ECOC predictor constructed from Naive Bayes PiCs effectively collapses all predictions for these four classes into *other*. This is fortuitous, since the distribution of test documents, reflected in Figure 7b, was highly skewed towards *other*.

Why the spike at $n = 8$? This is where the codeword for *other* diverges from the other three, and so the code has to make a decision: does the document belong to the *other* class, or to one of {*project*, *staff*, *student*}? The $n = 8$ code does sometimes correctly identify *project* documents, but at a huge cost in mispredicting *other*, as shown in Figure 7b. The unusual behavior arises from an inconsistency in the distribution of labels in training and test datasets, along with a "lucky" choice of $\mathcal{C}$.

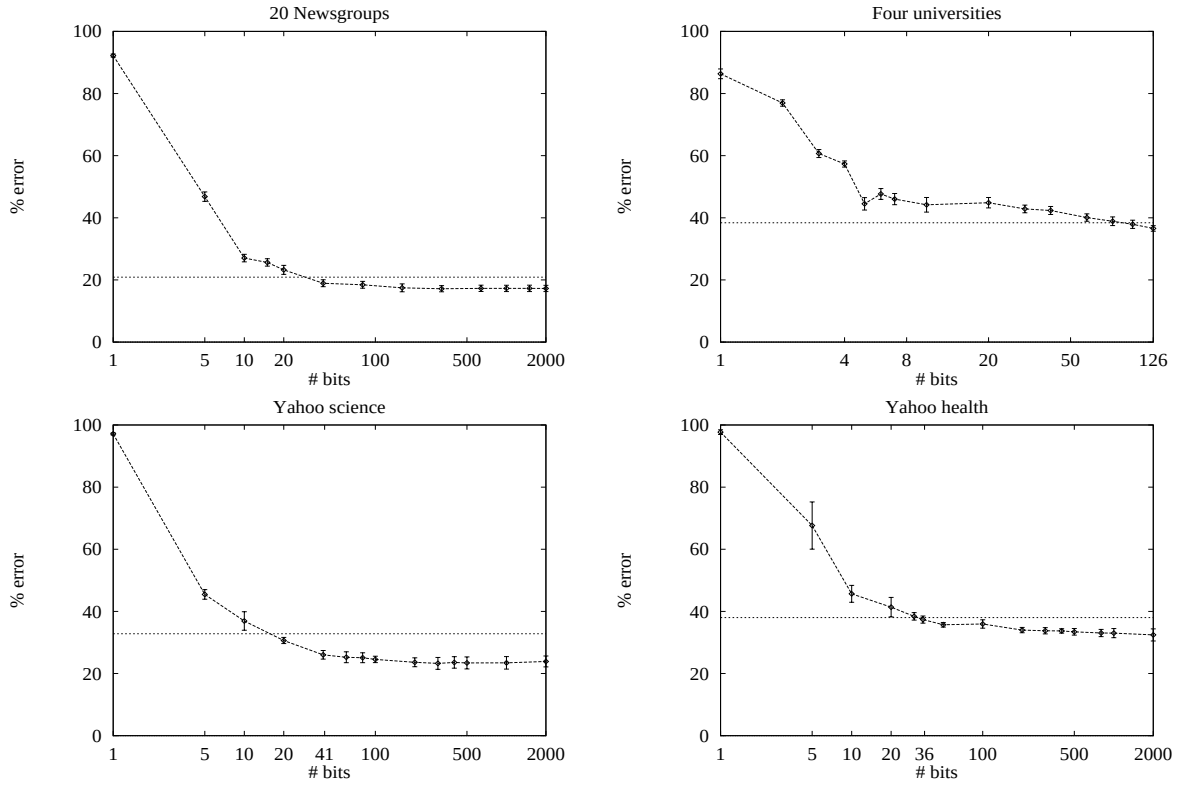

Figure 4: Performance of ECOC classification as a function of code size. Naive Bayes classifiers served as the PiCs. Each point reflects an average over five randomized training/test splits, and the bars measure the standard deviation over these trials. The horizontal line is the behavior of standard one-vs.-rest Naive Bayes. All points are averaged over five trials with a randomized $\mathcal{C}$ and randomized $3/4 - 1/4$ training/test split of the data.

3.4 Discussion

The results of the previous section suggest that up to a point, classifier performance improves with n. A simple calculation shows why this should be so.

Assume for the moment that the PiCs only output binary values, and the errors committed by any two PiCs are independent of one another. Denote by p_i the probability of error by the ith PiC, and let $\hat{p} \equiv \max_i p_i$. If the minimum distance of $\mathcal{C}$ is $\Delta_{\min}$, then classification is robust to any $\lfloor \Delta_{\min}/2 \rfloor$ or fewer errors, and so the probability of a correct classification, as a function of n, is

$$p(n) \geq \sum_{k=0}^{\lfloor \Delta_{\min}/2 \rfloor} \binom{n}{k} \hat{p}^k (1-\hat{p})^{n-k} \tag{4}$$

The quantity on the right—the first $\lfloor \Delta_{\min}/2 \rfloor$ terms of the binomial expansion of $p+(1-p)$—is monotonically increasing in $\Delta_{\min}$, which itself increases with n for a randomly-constructed code. Section 3.3 shows that in practice, $p(n)$ eventually plateaus, which means that the assumption that the errors are uncorrelated is false. This is hardly surprising: after all, the individual classifiers were trained on the same data. One would expect a correlation between, for instance, the second and third columns of the code presented in Section 3.1.

3.4.1 Relation to Naive Bayes

We have already seen that the one-vs.-rest strategy is a special case of ECOC classification. It is not difficult to see that the standard Naive Bayes approach is an implementation of ECOC classification. Notice that Naive Bayes is clearly a one-vs.-rest technique: predicting from among m classes requires constructing m

$\mathcal{S}^+$	top-ranked words
comp.graphics, comp.windows.x, rec.autos, rec.sport.hockey, sci.electronics, soc.religion.christian, talk.politics.guns talk.politics.mideast	LEAFS ISTANBUL OTTOMAN NHL BRUINS XLIB WIDGET KARABAKH R5 JPEG OPENWINDOWS SIGGRAPH ISRAELIS HOMICIDES GEICO GLOCK GUERILLAS MR2 AUTOS SAAB SHOTGUNS AUTOMATICS BUICK REVOLVER SPLINE ROTORS SELFHATING SOUL MOSQUE HUMILITY
comp.sys.mac.hardware, comp.windows.x, misc.forsale, rec.autos, rec.sport.baseball, sci.crypt, sci.med, soc.religion.christian, talk.politics.misc	PITCHER HIV NSA ESCROW R5 CIPHER WOLVERINE ALOMAR CRYPTOGRAPHIC KINSEY R4 PITCHING SPIDERMAN DENNING MUSTANG IMAKE BAGGED ANTIBIOTICS CIPHERTEXT RSA INFECTIONS POWERBOOK MIGRAINE QUACKS CORVETTE TBIRD BOLSHEVIK LEXUS SHALALA CONGREGATIONS

Table 3: Summarizing two Naive-Bayes PiCs learned from the 20 newsgroups dataset. The first column gives the labels comprising the positive examples, and the second column lists a selection of the top-ranking words, sorted by mutual information with the superclass.

label	coding
course	10101010
department	01100110
faculty	00011110
other	00000001
project	00000000
staff	00000000
student	00000000

Table 4: The first 8 bits of the code for the 4-universities/non-random partition experiment. This subset of the code doesn't distinguish among *project*, *staff*, and *student*, and the first four bits doesn't distinguish *other* from this group, either.

classifiers (each consisting of a prior $p(y_i)$ and a class-specific distribution $p(w \mid y_i)$), and selecting a label via (3). But this just amounts to using as a code $\mathcal{C}$ the m by m identity matrix, and then applying Algorithm 2 using an L_p norm.

3.4.2 Relation to k-nearest neighbor

A popular approach to text classification, particularly competitive for very large and sparse datasets, is k-nearest neighbor (kNN). kNN relies on a map $\psi : x \to \vec{v}$ from documents x to N-dimensional vectors $\vec{v}$. The entries of the latter may be word counts or, more generally, a list of feature values. A kNN classifier stores the images of all training set documents in a database $\mathcal{V} = \{\vec{v}_1, \vec{v}_2, \ldots\}$. To classify an unlabeled document x, kNN finds the k vectors in $\mathcal{V}$ closest to $\psi(x)$, and takes a weighted vote of their labels.

kNN and ECOC have some superficial similarities. Both use for classification a data structure consisting of a set of vectors, and both search this data structure using a nearest-neighbor algorithm, linear in the size of the data structure. One distinction—of particular importance when the size of the training set becomes large—is that while ECOC's data structure consists of a single vector for each label, kNN must store a vector for each document in the training set.

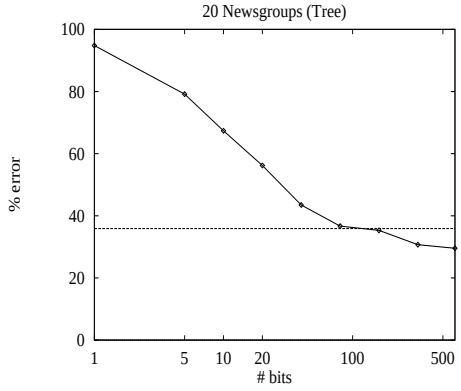

Figure 5: Performance of ECOC classification as a function of code size, for a decision tree PiC with a Bernoulli event model which takes no account of multiple appearances of a word in a document. Each point reflects a single trial using a randomized training/heldout/test partition of the 20 newsgroups collection. (The heldout data served to smooth the distributions at the nodes of the tree.) The horizontal line is the one-vs.-rest decision tree performance.

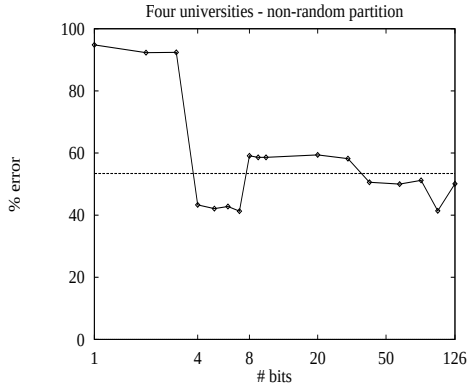

Figure 6: Accuracy against code size for a single trial of a more "honest," non-random partition of the four universities dataset. As before, the PiC is a Naive Bayes classifier. Notable is the behavior of the plot between $n = 4$ and $n = 8$.

4 Document retrieval: a source-channel framework

This section proposes a new probabilistic approach to information retrieval based upon the ideas and methods of statistical machine translation. The central ingredient in this approach is a statistical model of how a user might distill or "translate" a given document into a query. To assess the relevance of a document to a user's query, we estimate the probability that the query would have been generated as a translation of the document, and factor in the user's general preferences in the form of a prior distribution over documents. We propose a simple, well motivated model of the document-to-query translation process, and describe an algorithm for learning the parameters of this model in an unsupervised manner from a collection of documents. As we show, one can view this approach as a generalization and justification of the "language modeling" strategy recently proposed by Ponte and Croft [32]. On a series of experiments, a prototype translation-based retrieval system significantly outperforms conventional retrieval techniques. This prototype system is but a skin-deep implementation of the full translation-based approach, and as such only begins to tap the full potential of translation-based retrieval.

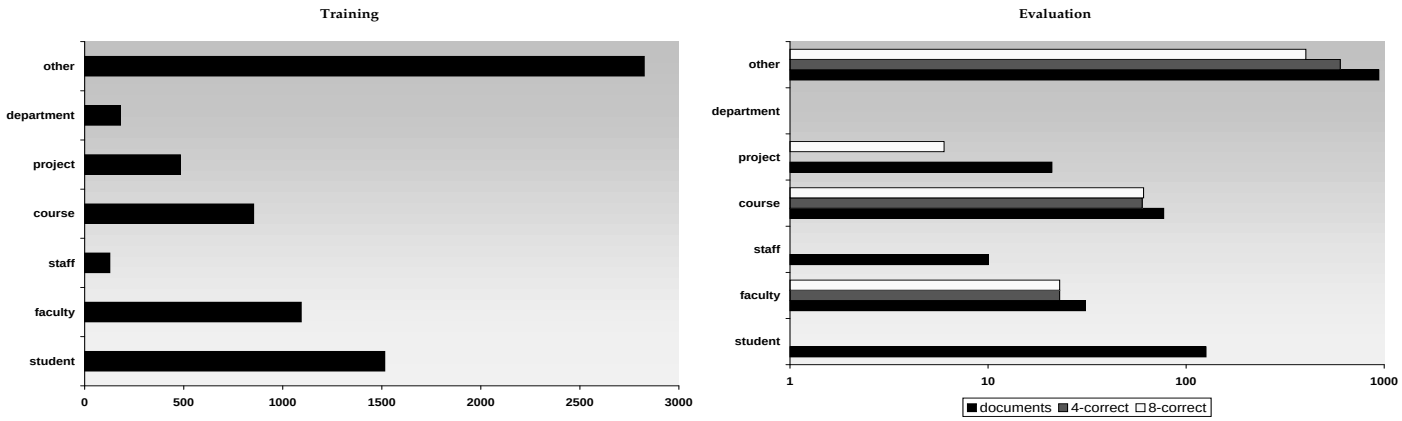

Figure 7: Left (a): Distribution of labels in training data for a non-random partition of the 4-universities collection. Right (b): Test distribution in test data, and the per-label accuracy for the 4- and 8-bit classifiers. The x-axis in (b) is on a logarithmic scale for clarity. The difference in training and test distributions, along with a "lucky" choice of $\mathcal{C}$, leads to anomalous behavior: the four-bit code strongly outperforms the 8-bit code.

4.1 Background

When a user formulates a query to a retrieval system, what he is really doing is distilling an information need into a succinct query. In this work, we take the view that this distillation is a form of translation from one language to another: from documents, which contain the normal superfluence of textual fat and connective tissue such as prepositions, commas and so forth, to queries, comprised of just the skeletal index terms that characterize the document.

We take this view not because it is an accurate model of how a user decides what to ask of an information retrieval system, but because it turns out to be a useful expedient. By thinking about retrieval in this way, we can formulate tractable mathematical models of the query generation and retrieval process, models that can be implemented in a quite straightforward way and that exhibit promising empirical behavior.

Viewing document retrieval as a problem in translation may seem rather fanciful. But in fact, from this perspective, the task of retrieval is just a matter of inverting the translation: given a query, find the document(s) in the collection most likely to translate to the query. Moreover, the translation perspective offers another compelling argument in its favor: one can dispense with many of the *ad hoc* garlands common and critical to modern high-performance retrieval systems, such as query expansion and term weighting, since they are built in to the translation framework.

We begin by providing more detail on the source-channel model of document retrieval introduced in Section 2. In formulating a query to a retrieval system, a user begins with an information need. We view this need as an fragment of an "ideal document"—a perfect fit for the user, but almost certainly not present in the retrieval system's collection of documents. From this text, the user selects a group of identifying terms. In the context of traditional IR, one could view this group of terms as akin to an expanded query. The user then formulates a query from this group of terms by removing duplicates and replacing some terms with related terms: replacing PONTIFF with POPE, for instance.

Summarizing the model of query generation,

1. The user has an information need $\Im$.
2. From this need, he generates an ideal document fragment $\mathbf{d}_\Im$.
3. He selects a set of key terms from $\mathbf{d}_\Im$, and generates a query $\mathbf{q}$ from this set.

One can view this imaginary process of query formulation—from an ideal document $\mathbf{d}_\Im$ to a query $\mathbf{q}$—as a corruption of the ideal document. In this setting, the task of a retrieval system is to find those documents

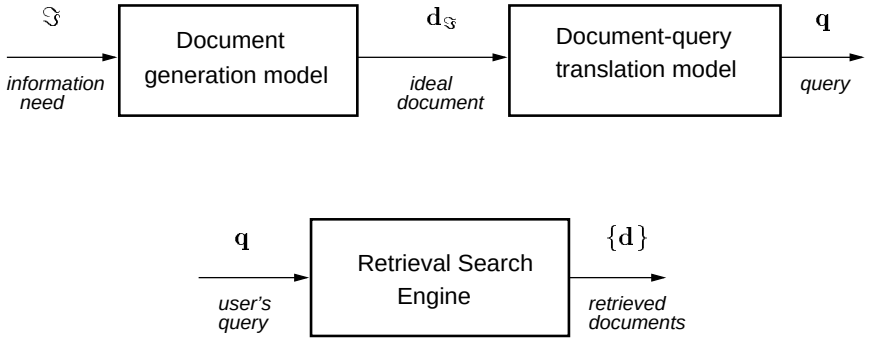

Figure 8: Model of query generation (above) and retrieval (below)

most similar to $\mathbf{d}_\Im$. In other words, retrieval is the task of finding, among the documents comprising the collection, likely preimages of the user's query. Figure 8 depicts this model of retrieval in a block diagram.

Not surprisingly, Figure 8 suggests an information-theoretic perspective. One can view the information need $\Im$ as a message that gets corrupted as the user $\mathcal{U}$ distills it into a query $\mathbf{q}$. That is, the query-formulation process represents a noisy channel, corrupting the information need just as a telephone cable corrupts the data transmitted by a modem. Given $\mathbf{q}$ and a model of the channel—how an information need gets corrupted into a query—the retrieval system's task is to identify those documents $\mathbf{d}$ that best satisfy the information need of the user.[3]

More precisely, the retrieval system's task is to find the *a posteriori* most likely documents given the query; that is, those $\mathbf{d}$ for which $p(\mathbf{d} \mid \mathbf{q}, \mathcal{U})$ is highest. By Bayes' law,

$$p(\mathbf{d} \mid \mathbf{q}, \mathcal{U}) = \frac{p(\mathbf{q} \mid \mathbf{d}, \mathcal{U})\; p(\mathbf{d} \mid \mathcal{U})}{p(\mathbf{q} \mid \mathcal{U})}\,. \tag{5}$$

Since the denominator $p(\mathbf{q} \mid \mathcal{U})$ is fixed for a given query and user, we can ignore it for the purpose of ranking documents, and define the relevance $\rho(\mathbf{d}; \mathbf{q})$ of a document to a query as

$$\rho(\mathbf{d}; \mathbf{q}) = \underbrace{p(\mathbf{q} \mid \mathbf{d}, \mathcal{U})}_{\text{query-dependent}} \quad \underbrace{p(\mathbf{d} \mid \mathcal{U})}_{\text{query-independent}}\,. \tag{6}$$

Equation (6) highlights the decomposition of relevance into two terms: first, a query-dependent term measuring the relevance of $\mathbf{d}$ to $\mathbf{q}$, and second, a query-independent or "prior" term, measuring the quality of the document according to the user's preferences. Though in this work we take the prior term to be uniform over all documents, ultimately the prior will be very important for improved performance, and for allowing the system to adapt to the user's needs and interests. At the very least, the document prior can be used to discount short documents, or perhaps documents in a foreign language.

Section 4.4 contains a detailed formulation of a simple $p(\mathbf{q} \mid \mathbf{d})$ model, but here we will briefly outline the strategy for constructing the model. The strategy requires a corpus of $(\mathbf{d}, \mathbf{q})$ pairs, with each pair consisting of a query and a document relevant to the query. Section 4.5 describes how, given this data, one can construct a translation model $p(\mathbf{q} \mid \mathbf{d})$, which assigns a probability to the event that $\mathbf{q}$ is a distillation of (a translation of) $\mathbf{d}$. Given such a model and a new query $\mathbf{q}'$, assigning relevance judgments is a matter of computing $p(\mathbf{q}' \mid \mathbf{d})$ for each $\mathbf{d} \in \mathcal{C}$.

Ultimately, as mentioned earlier, document retrieval systems must be sophisticated enough to handle polysemy and synonymy—to know, for instance, that PONTIFF and POPE are related terms. The field of statistical translation concerns itself with how to mine large text databases to automatically discover such semantic relations. Brown *et al.* [11, 13] showed, for instance, how a system can "learn" to associate French terms with their English translations, given only a collection of bilingual French/English sentences. We shall demonstrate how, in a similar fashion, an IR system can, from from a collection of documents, automatically "learn" which terms are related, and exploit these relations to better find and rank the documents it returns to the user.

[3] We use in this section the convention that boldface roman letters refer to collections of words such as documents or queries, while italic roman letters refer to individual terms. Thus $p(q \mid \mathbf{d})$ refers to the probability of generating a single query word from an entire document $\mathbf{d}$.

4.2 Related work

In modeling how a document is distilled into a query, our approach bears some similarity to a longstanding concern in the field of library science: finding a concise group of index terms that characterize a document. Towards this end, Bookstein and Swanson [7] observed, in proposing their 2-Poisson model, that the appearance of content terms in a document follows one of two distributions: POPE, for instance, follows a Poisson distribution with a higher mean in documents about the Pope, and a distribution with a lower mean in other documents. So term frequency information could be useful for an algorithm whose responsibility is to automatically compile a list of index terms for documents. Though an important step in both indexing and retrieval, this model is imperfect (as the authors concede) in that there exist shades of "aboutness"—not all documents are either about the Pope or only peripherally about the Pope. Moreover, the family of n-Poisson models have to date achieved only limited empirical success [32].

In the same paper, Bookstein and Swanson also propose a mathematical model to account for the fact that over a heterogeneous collection of documents, content terms (like LASER) tend to occur in clusters, while non-content terms (like OBTAIN) typically occur more uniformly. In a more detailed study of this phenomenon, Robertson and Sparck Jones [34] used statistical techniques to analyze the distribution of search terms, with an eye towards increasing the performance of document retrieval systems by automatically assigning different "relevancy" weights to the terms in a query. Refinements of this work by many parties over the past few decades have led to the popular *tfidf* weighting scheme [37], a central ingredient in the search engines of several multi-billion dollar companies.

Robertson and Sparck Jones used these relevance weights as part of a probabilistic approach to retrieval: given a query and a set of documents, estimate for each document the probability that the document is relevant to the query. This retrieval framework—predicting whether the document is relevant to the query—is a popular one, appearing in other probabilistic approaches to IR as well [16]. In contrast, the approach we take is to evade the notion of "relevance" altogether, replacing it with an information-theoretic perspective that asks whether the document could have given rise to the query through a process of distillation. While the parameters of *tfidf* and Okapi systems [35] can be manually adjusted to improve performace, our approach is to construct automatically trainable models that can be fit from data and adapted to a user's interaction with the system.

A quite different probabilistic approach to retrieval from the relevancy-prediction strategy appears in the INQUERY system, under development at University of Massachusetts [39]. As with the approach presented herein, the INQUERY system distinguishes between a user's (hidden) information need and his query. INQUERY uses a Bayesian network to predict whether the retrieved documents satisfy the information need of the user, as expressed in the query.

Perhaps closest in spirit to the present work is the language modeling approach introduced recently by Ponte and Croft [32, 31]. To each document in the collection, they associate a probability distribution $p(\cdot \mid \mathbf{d})$ over terms; they call this distribution a "language model" to accord with terminology developed in the field of speech recognition. The probability of a term t given a document is related to the frequency of t in the document. The probability of a query $\mathbf{q} = q_1, q_2, \ldots q_m$ is just the product of the individual term probabilities, $p(\mathbf{q} \mid \mathbf{d}) = \prod_i p(q_i \mid \mathbf{d})$. The relevance of a document $\mathbf{d}$ to a query $\mathbf{q}$ is presumed to be monotonically related to $p(\mathbf{q} \mid \mathbf{d})$.

The language modeling approach represents a novel and theoretically motivated approach to retrieval, which Ponte and Croft demonstrated to be effective. However, their framework does not appear general enough to handle the important issues of synonymy and polysemy.

4.3 Statistical translation

Besides its intellectual pedigree in information retrieval, the roots of the present work lie in the field of statistical machine translation (MT). Automatic translation by computer was first contemplated by Warren Weaver when modern computers were in their infancy [40], but received serious attention only in the past decade. In particular, the form of the mathematical model that will appear in Section 4.4 bears a resemblance to one introduced in [12] in the context of machine translation.

The central problem of statistical MT is to build a system that automatically learns how to translate text, by inspecting a large corpus of sentences in one language along with their translations into another language. The Candide system, an experimental project at IBM Research in the early 1990s, employed for

this purpose the recorded proceedings of the Canadian Parliament, which happen to be maintained in both English and French.

As hinted at above, language translation has a convenient restatement in information-theoretic terms. We illustrate with the example of a French-to-English translation system that has just been presented with a French sentence to translate. Imagine that the person who generated this French sentence was in fact thinking of an English sentence, which was somehow "corrupted" (between the user's conceiving of it, and presenting it to the translation system) into the given French sentence. The goal of the translation system is to recover the original English sentence. In performing this inversion, the translation system has at its disposal not just the input French sentence, but also a model of how English sentences are "corrupted" into French sentences and a model of well-formed English sentences, both learned automatically from the bilingual corpus.

The reader may recognize this framework as analogous to how we have earlier posed the document retrieval problem. The original English sentence is akin to the ideal document representing the user's information need; the French sentence is like the query. One important difference between the two settings is that the search for the optimal input to the channel is, in the case of retrieval, restricted to documents in the collection, whereas a translation system considers (at least in principle) all English sentences.

Of course, the source-channel framework is only a restatement of the problem, not a solution. Still missing is the parametric form of the channel model, a method for learning the parameters of this model from a corpus, and a way to apply this model to the task of translating an input French sentence. It would take us too far afield to offer any detail on these steps, and we refer the reader to Brown *et al.* [11, 13] for details on all three issues. Instead we briefly describe the most primitive of the IBM statistical translation models. This model contains a *translation probability* $t(f \mid e)$ for each English word e translating to each French word f. The probability that an English sentence $\mathbf{e} = \{e_1, e_2, \ldots\}$ translates to a French sentence $\mathbf{f} = \{f_1, f_2 \ldots\}$ is calculated as

$$p(\mathbf{f} \mid \mathbf{e}) = \gamma \sum_{\mathbf{a}} \prod_{j=1}^{m} t(f_j \mid e_{a_j})$$

where γ is a normalizing factor. The hidden variable in this model is the *alignment* $\mathbf{a}$ between the French and English words: $a_j = k$ means that the kth English word translates to the jth French word.

Brown *et al.* [13] propose a series of increasingly complex and powerful statistical models of translation, of which this model is the first, and appropriately called *Model 1*.

4.4 A Simple Model of Document-Query Translation

In this section we introduce a simple model of how documents are distilled or "translated" into queries. Suppose that an information analyst is given a news article and asked to quickly generate a list of a few words to serve as a rough summary of the article's topic. As the analyst rapidly skims the story, he encounters a collection of words and phrases. Many of these are rejected as irrelevant, but his eyes rest on certain key terms as he decides how to render them in the summary. For example, when presented with an article about Pope John Paul II's visit to Cuba in 1998, the analyst decides that the words PONTIFF and VATICAN can simply be represented by the word POPE, and that CUBA, CASTRO and ISLAND can be collectively referred to as CUBA.

To represent this process in terms of a statistical model, we make the assumption, clearly invalid, that the analyst generates a list of words by making several *independent* translations of the document $\mathbf{d}$ into a single query term q, in the following manner. First, the analyst chooses a word w at random from the document. He chooses this word according to a distribution $l(w \mid \mathbf{d})$ that we call the *document language model*. Next, he translates w into the word or phrase q according to a *translation model*, with parameters $t(q \mid w)$. Thus, the probability of choosing q as a representative of the document $\mathbf{d}$ is

$$p(q \mid \mathbf{d}) = \sum_{w \in \mathbf{d}} l(w \mid \mathbf{d})\, t(q \mid w)\,.$$

We assume that the analyst repeats this process n times, where n is chosen according to the *sample size model* $\phi(n \mid \mathbf{d})$, and that the resulting list of words is filtered to remove duplicates before it is presented as the summary, or query, $\mathbf{q} = q_1, q_2, \ldots q_m$.

In order to calculate the probability that a particular query $\mathbf{q}$ is generated in this way, we need to sum over all sample sizes n, and consider that each of the terms q_i may have been generated multiple times. Thus, the process described above assigns to $\mathbf{q}$ a total probability

$$\begin{aligned} p(\mathbf{q} \mid \mathbf{d}) &= \sum_n \phi(n \mid \mathbf{d}) \sum_{n_1>0} \cdots \sum_{n_m>0} \binom{n}{n_1 \cdots n_m} \prod_{i=1}^{m} p(q_i \mid \mathbf{d})^{n_i} \\ &= \sum_n \phi(n \mid \mathbf{d}) \sum_{n_1>0} \cdots \sum_{n_m>0} \binom{n}{n_1 \cdots n_m} \prod_{i=1}^{m} \left(\sum_w l(w \mid \mathbf{d})\, t(q_i \mid w) \right)^{n_i} . \end{aligned}$$

In spite of its intimidating appearance, this expression can be calculated efficiently using simple combinatorial identities and dynamic programming techniques. Instead of pursuing this path, we will assume that the number of samples n is chosen according to a Poisson distribution with mean $\lambda(\mathbf{d})$:

$$\phi(n \mid \mathbf{d}) = e^{-\lambda(\mathbf{d})} \frac{\lambda(\mathbf{d})^n}{n!} .$$

Under this assumption, the above sum takes on a much more friendly appearance:

$$p(\mathbf{q} = q_1, \ldots, q_m \mid \mathbf{d}) = e^{-\lambda(\mathbf{d})} \prod_{i=1}^{m} \left(e^{\lambda(\mathbf{d})\, p(q_i \mid \mathbf{d})} - 1 \right) . \tag{7}$$

This formula shows that the probability of the query is given as a product of terms. Yet the query term translations are *not* independent, due to the process of filtering out the generated list to remove duplicates.

Just as the most primitive version of IBM's translation model takes no account of the subtler aspects of language translation, including the way word order tends to differ across languages, so our basic IR translation approach is but an impressionistic model of the relation between queries and documents relevant to them. Since IBM called their most basic scheme *Model 1*, we shall do the same for this rudimentary retrieval model.

Both our Model 1 for document-to-query translation and IBM Model 1 for English-French natural language translation contain only two basic types of parameters: string length probabilities, and word translation probabilities. However, the form of the models is qualitatively and mathematically different. Document-query translation requires a *distillation* of the document, while natural language translation will tolerate little being thrown away.

In fact, our Model 1 was inspired by another IBM statistical translation model, but one that was designed for modeling a bilingual dictionary [12]. In this model, one imagines a dictionary entry being constructed by a lexicographer collecting translations of an English word or phrase $\mathbf{e}$. After gathering a number of independent translations, the lexicographer removes duplicates to form a dictionary entry for $\mathbf{e}$. Our basic model and training algorithm is an adaptation of this scheme.

4.4.1 The simplest case: word-for-word translation

The simplest version of the above model, which we will distinguish as *Model 0*, is one where each word w can be translated only as itself; that is, the translation probabilities are "diagonal":

$$t(q \mid w) = \begin{cases} 1 & \text{if } q = w \\ 0 & \text{otherwise} . \end{cases}$$

Under this model, the query terms are chosen simply according to their frequency of occurrence in the document (or some smoothed version of that frequency). As a further simplification, let us fix the average number of samples to be a constant λ independent of the document $\mathbf{d}$, and suppose that the expected number of times a query word is drawn is less than one, so that $\max_i \lambda l(q_i \mid \mathbf{d}) < 1$. Then to first order, the probability assigned to the query is a constant times the product of the language model probabilities:

$$p(\mathbf{q} = q_1, \ldots, q_m \mid \mathbf{d}) \approx e^{-\lambda}\, \lambda^m \prod_{i=1}^{m} l(q_i \mid \mathbf{d}) \tag{8}$$

Since the mean λ is fixed for all documents, the document that maximizes the righthand side of the above expression is that which maximizes the product $\prod_{i=1}^{m} l(q_i \mid \mathbf{d})$. This is precisely the value assigned to the query in what Ponte and Croft (1998) call the "language modeling approach."

4.5 Building the System

We now describe an implementation of Model 1 as described in the previous section, and its application to TREC data. The key ingredient in Model 1 is the collection of translation probabilities $t(q \mid w)$. But how are we to obtain these probabilities? The statistical translation strategy is to learn these probabilities from an aligned bilingual corpus of translated sentences. Ideally, we should have a collection of query/document pairs to learn from, obtained by human relevance judgments. But a collection of such data of the size needed to estimate parameters for general queries is difficult to come by.

Lacking a large corpus of queries and their associated relevant documents, we decided to tease out the semantic relationships among words by generating *synthetic queries* for a large collection of documents, and training the Model 1 translation probabilities on this synthetic data. To explain the rationale for this scheme, we return to our fictitious information analyst, and recall that when presented with a document $\mathbf{d}$, he will tend to select terms that are suggestive of the content of the document. Suppose now that he himself selects an arbitrary document $\mathbf{d}$ from a database $\mathcal{D}$, and asks us to guess, based only upon his summary $\mathbf{q}$, which document he had chosen. The amount by which we are able to do better, on average, than randomly guessing a document from $\mathcal{D}$ is the *mutual information* $I(D;Q) = H(D) - H(D \mid Q)$ between the random variables representing his choice of document D and query Q. Here $H(D)$ is the entropy in the analyst's choice of document, and $H(D \mid Q)$ is the conditional entropy of the document given the query. If he is playing this game cooperatively, he will generate queries for which this mutual information is large.

With this game in mind, we took a collection of TREC documents $\mathcal{D}$, and for each document $\mathbf{d}$ in the collection, we weighted and ranked its words $w \in \mathbf{d}$ according to their mutual information statistic

$$I(w, \mathbf{d}) = p(w, \mathbf{d}) \log \frac{p(w \mid \mathbf{d})}{p(w \mid \mathcal{D})} .$$

Here $p(w \mid \mathbf{d})$ is the probability of the word in the document, and $p(w \mid \mathcal{D})$ is the probability of the word in the collection at large. (While the quantity $I(w, \mathbf{d})$ is typically positive, we could encounter negative values and so need to scale these statistics appropriately.) We then drew $n \sim Poisson(\lambda)$ random samples from the document according to this distribution.

Given this synthetic corpus of documents and queries, we fit the translation probabilities of Model 1 to it using the EM algorithm [17], run only for three iterations as a simple means to avoid overfitting. Space limitations prevent us from explaining the details of the training process, but some of these can be found in the papers [13, 12] which describe similar models. A sample of the resulting translation probabilities, when trained on the $Associated\ Press$ (AP) portion of the TREC volume 3 corpus, are shown in Table 5. In this table, a document word is shown together with the ten most probable query words that it will translate to according to the model. The probabilities in these particular tables are among the 47,065,200 translation probabilities that were trained for our 132,625 word vocabulary. They were estimated from a corpus obtained by generating five synthetic mutual information queries for each of the 78,325 documents in the collection.

For statistical models of this form, *smoothing* or interpolating the parameters away from their maximum likelihood estimates is crucial. We used a simple linear mixture of the background unigram model and the EM-trained translation model:

$$\begin{aligned} p_\alpha(q \mid \mathbf{d}) &= \alpha\, p(q \mid \mathcal{D}) + (1-\alpha)\, p(q \mid \mathbf{d}) \\ &= \alpha\, p(q \mid \mathcal{D}) + (1-\alpha) \sum_{w \in \mathbf{d}} l(w \mid \mathbf{d})\, t(q \mid w) . \end{aligned}$$

The weight was empirically set to $\alpha = 0.05$ by optimizing performance on a different dataset: a portion of the 1998 TREC *Spoken Document Retrieval* (SDR) data. The models for the baseline language modeling approach, or Model 0, were also smoothed using linear interpolation:

$$l_\gamma(w \mid \mathbf{d}) = \gamma\, p(w \mid \mathcal{D}) + (1-\gamma)\, l(w \mid \mathbf{d}) .$$

This mixture weight was simply fixed at $\gamma = 0.1$. The Poisson parameter for the sample size distribution was fixed at $\lambda = 15$, independent of the document. No adjustment of any parameters, other than that for the unsupervised EM training the translation probabilities, was carried out on the TREC volume 3 data that we ran our evaluation on.

q	$t(q \mid w)$
PONTIFF	0.502
POPE	0.169
PAUL	0.065
JOHN	0.035
VATICAN	0.033
II	0.028
VISIT	0.017
PAPAL	0.010
CHURCH	0.005
FLIGHT	0.004

w = PONTIFF

q	$t(q \mid w)$
DEFEND	0.676
TRIAL	0.015
CASE	0.012
COURT	0.011
CHARGE	0.011
JUDGE	0.010
ATTORNEY	0.009
CONVICT	0.007
PROSECUTOR	0.006
ACCUSE	0.006

w = DEFEND

q	$t(q \mid w)$
WILDLIFE	0.705
FISH	0.038
ACRE	0.012
SPECIES	0.010
FOREST	0.010
ENVIRONMENT	0.009
HABITAT	0.008
ENDANGERED	0.007
PROTECTED	0.007
BIRD	0.007

w = WILDLIFE

q	$t(q \mid w)$
SOLZHENITSYN	0.319
CITIZENSHIP	0.049
EXILE	0.044
ARCHIPELAGO	0.030
ALEXANDER	0.025
SOVIET	0.023
UNION	0.018
KOMSOMOLSKAYA	0.017
TREASON	0.015
VISHNEVSKAYA	0.015

w = SOLZHENITSYN

q	$t(q \mid w)$
CARCINOGEN	0.667
CANCER	0.032
SCIENTIFIC	0.024
SCIENCE	0.014
ENVIRONMENT	0.013
CHEMICAL	0.012
EXPOSURE	0.012
PESTICIDE	0.010
AGENT	0.009
PROTECT	0.008

w = CARCINOGEN

q	$t(q \mid w)$
ZUBIN_MEHTA	0.248
ZUBIN	0.139
MEHTA	0.134
PHILHARMONIC	0.103
ORCHESTRA	0.046
MUSIC	0.036
BERNSTEIN	0.029
YORK	0.026
END	0.018
SIR	0.016

w = ZUBIN

q	$t(q \mid w)$
IBM	0.674
COMPUTER	0.042
MACHINE	0.022
ANALYST	0.012
SOFTWARE	0.011
WORKSTATION	0.007
STOCK	0.006
SYSTEM	0.006
BUSINESS	0.005
MARKET	0.005

w = IBM

q	$t(q \mid w)$
EVEREST	0.439
CLIMB	0.057
CLIMBER	0.045
WHITTAKER	0.039
EXPEDITION	0.036
FLOAT	0.024
MOUNTAIN	0.024
SUMMIT	0.021
HIGHEST	0.018
REACH	0.015

w = EVEREST

q	$t(q \mid w)$
WHITTAKER	0.535
CLIMBER	0.048
EVEREST	0.032
CLIMB	0.023
EXPEDITION	0.018
GARBAGE	0.015
CHINESE	0.015
PEACE	0.015
COOPER	0.013
1963	0.012

w = WHITTAKER

Table 5: Sample translation probabilities

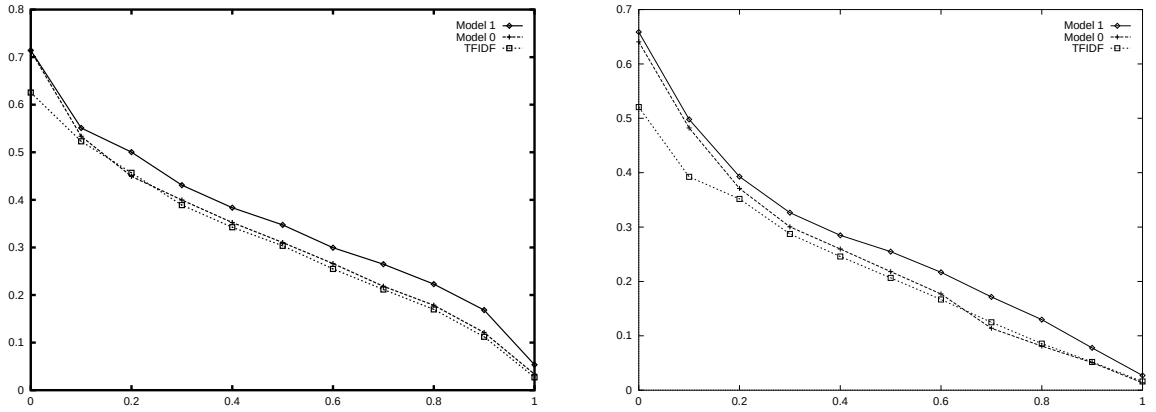

Figure 9: Precision-recall curves on TREC data. The plots compares the performance of Model 1 to the baseline Model 0 on AP data (left) and SJMN data (right) when ranking documents for queries formulated from the concept fields for topics 51–100.

4.6 Experimental Results on TREC Data

In this section we summarize the quantitative results of using Model 1, as described in the previous two sections, to rank documents for a set of queries. Though not a real-time computation, calculating the relevance score $\rho(\mathbf{d};\mathbf{q})$ for each query occurs quickly enough on modern-day equipment to obviate the need for a "fast match" to throw out obviously irrelevant documents.

Our experiments fall into three categories. First, we report on a series of experiments carried out on the AP portion of the TREC data from volume 3 for both the concept and title fields in topics 51–100. The concept field queries comprise a set of roughly 20 keywords, while the title fields are much more succinct—typically not more than four words. Next, we tabulate a corresponding series of experiments carried out on the *San Jose Mercury News* (SJMN) portion of the data, also evaluated on topics 51–100. Finally, we present results on the *Spoken Document Retrieval* (SDR) track of the 1998 TREC evaluation, a relatively small collection of about 2800 transcripts of news broadcasts.

Precision-recall curves for the AP and SJMN data, generated from the output of the TREC evaluation software, appear in Figure 9. The baseline curve in this plot is the result of using Model 0 to score the documents, using only word-for-word translations. In this approach, documents receive high relevance scores by containing terms appearing in the query. Model 1 improves the average precision over this baseline by 15.9% on the AP data, and by 10.5% on the SJMN data. The R-precision improves by 11.0% on the AP data and by 6.5% on the SJMN data. The actual numbers are tabulated in Table 6.

These plots also show the performance of the *tfidf* measure using Robertson's *tf* score, as described by Ponte in [31]. Although Ponte and Croft report an appreciable improvement of 8.7% in average precision and 6.2% in R-precision on these same queries [32, 31], evaluated however on the *entire* volume 3 collection, we see only a small difference of 3.5% and 0.3% on the AP portion of the data. We expect that this can be attributed to our simple method of linear interpolation with a fixed weight, and underscores the need for smoothing when working with these language models.

As discussed earlier, overfitting during EM training is a concern because of the manner in which the synthetic queries are generated. In Figure 4.6 we show the performance of Model 1 on the AP data when the probabilities are trained for two and three iterations. To study the effects of query length on the models, we also scored the documents for the title fields of topics 51–100, where the average query length is only 2.8 words. As Figure 4.6 reveals, the precision-recall curves are qualitatively different, showing a degradation in performance in the high-precision range. Overall, Model 1 achieves a smaller improvement of 5.2% in average precision and 2.7% in R-precision on these short queries; the numbers are tabulated in Table 4.6.

In Figure 11 two precision-recall plots for the SDR task are given. The left plot shows the improvement of Model 1 over the model having only the trivial word-for-word translation probabilities. There is an improvement of 17.2% in average precision and 12.9% in R-precision. The right plot compares Model 0 to the result of ranking documents according to the probability $\prod_{i=1}^{m} l(q_i \mid \mathbf{d})$ for the same language model, as in Ponte and Croft's method. There is very little difference in the results, showing that equation (8) is indeed a good approximation. These same data are tabulated in Table 4.6.

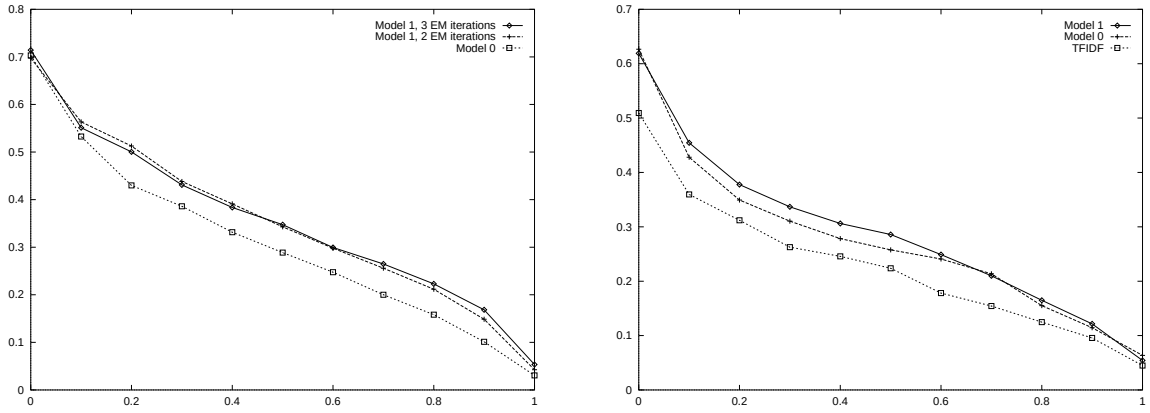

Figure 10: Precision-recall curves on the AP portion of the TREC data. The left plot shows the discrepancy between two and three EM iterations of training Model 1. The right plot shows the performance of Model 0 and Model 1 on the short (average 2.8 words/query) queries obtained from the title field of topics 51–100.

AP	Model 0	Model 1	%Δ
Relevant:	5845	5845	—
Rel.ret.:	5845	5845	0.0
Precision:			
at 0.00	0.7031	0.6974	-0.8
at 0.10	0.5327	0.5634	+5.8
at 0.20	0.4299	0.5127	+19.3
at 0.30	0.3861	0.4381	+13.5
at 0.40	0.3316	0.3911	+17.9
at 0.50	0.2887	0.3429	+18.8
at 0.60	0.2475	0.2980	+20.4
at 0.70	0.2001	0.2559	+27.9
at 0.80	0.1582	0.2117	+33.8
at 0.90	0.1012	0.1486	+46.8
at 1.00	0.0306	0.0426	+39.2
Avg.:	0.2937	0.3405	+15.9
Precision at:			
5 docs:	0.5404	0.5404	-0.0
10 docs:	0.5000	0.5170	+3.4
15 docs:	0.4596	0.4922	+7.1
20 docs:	0.4447	0.4723	+6.2
30 docs:	0.4121	0.4362	+5.8
100 docs:	0.2960	0.3330	+12.5
200 docs:	0.2283	0.2600	+13.9
500 docs:	0.1398	0.1569	+12.2
1000 docs:	0.0866	0.0959	+10.7
R-Precision:	0.3153	0.3499	+11.0

SJMN	Model 0	Model 1	%Δ
Relevant:	2322	2322	—
Rel.ret.:	2322	2322	0.0
Precision:			
at 0.00	0.6319	0.6586	+4.2
at 0.10	0.4891	0.4980	+1.8
at 0.20	0.3730	0.3928	+5.3
at 0.30	0.3023	0.3264	+8.0
at 0.40	0.2653	0.2848	+7.4
at 0.50	0.2271	0.2548	+12.2
at 0.60	0.1915	0.2169	+13.3
at 0.70	0.1244	0.1715	+37.9
at 0.80	0.0894	0.1297	+45.1
at 0.90	0.0569	0.0779	+36.9
at 1.00	0.0152	0.0270	+77.6
Avg.:	0.2349	0.2595	+10.5
Precision at:			
5 docs:	0.4750	0.4583	-3.6
10 docs:	0.3979	0.4021	+1.1
15 docs:	0.3528	0.3708	+5.1
20 docs:	0.3250	0.3385	+4.2
30 docs:	0.2903	0.3042	+4.8
100 docs:	0.1767	0.1948	+10.2
200 docs:	0.1203	0.1327	+10.3
500 docs:	0.0622	0.0687	+10.5
1000 docs:	0.0354	0.0386	+9.0
R-Precision:	0.2577	0.2745	+6.5

Table 6: Model 1 compared to the baseline system for queries constructed from the concept fields. These numbers correspond to the plots in Figure 9.

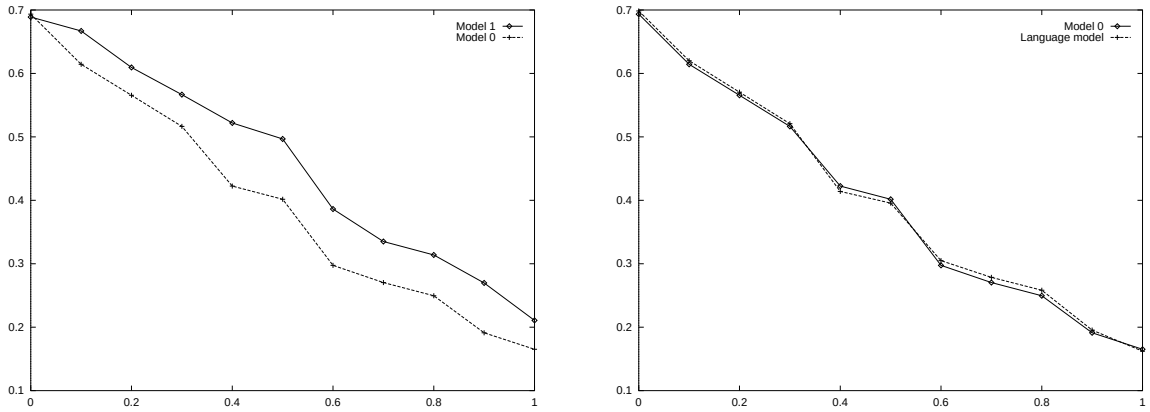

Figure 11: The left plot compares Model 1 to Model 0 on the SDR data. The right plot compares the same language model scored according to Model 0 and using the product $\prod_{i=1}^{m} l(q_i \mid \mathbf{d})$ demonstrating that the approximation in equation (8) is very good.

SDR	Model 0	Model 1	%Δ
Relevant:	390	390	—
Rel.ret.:	390	390	0.0
Precision:			
at 0.00	0.6934	0.6889	-0.7
at 0.10	0.6144	0.6670	+8.6
at 0.20	0.5656	0.6094	+7.7
at 0.30	0.5165	0.5666	+9.7
at 0.40	0.4225	0.5220	+23.6
at 0.50	0.4016	0.4967	+23.7
at 0.60	0.2975	0.3863	+29.8
at 0.70	0.2704	0.3351	+23.9
at 0.80	0.2495	0.3139	+25.8
at 0.90	0.1912	0.2697	+41.1
at 1.00	0.1652	0.2107	+27.5
Avg.:	0.3805	0.4458	+17.2
Precision at:			
5 docs:	0.4348	0.5217	+20.0
10 docs:	0.3783	0.4130	+9.2
15 docs:	0.3188	0.3565	+11.8
20 docs:	0.2761	0.3152	+14.2
30 docs:	0.2406	0.2638	+9.6
100 docs:	0.1030	0.1217	+18.2
200 docs:	0.0600	0.0700	+16.7
500 docs:	0.0303	0.0315	+4.0
1000 docs:	0.0159	0.0167	+5.0
R-Precision:	0.3753	0.4237	+12.9

SDR	Model 0	LM	%Δ
Relevant:	390	390	—
Rel.ret.:	390	390	0.0
Precision:			
at 0.00	0.6934	0.6985	+0.7
at 0.10	0.6144	0.6201	+0.9
at 0.20	0.5656	0.5705	+0.9
at 0.30	0.5165	0.5208	+0.8
at 0.40	0.4225	0.4140	-2.1
at 0.50	0.4016	0.3956	-1.5
at 0.60	0.2975	0.3049	+2.5
at 0.70	0.2704	0.2785	+3.0
at 0.80	0.2495	0.2582	+3.5
at 0.90	0.1912	0.1954	+2.2
at 1.00	0.1652	0.1622	-1.8
Avg.:	0.3805	0.3838	+0.9
Precision at:			
5 docs:	0.4348	0.4435	+2.0
10 docs:	0.3783	0.3870	+2.3
15 docs:	0.3188	0.3188	-0.0
20 docs:	0.2761	0.2848	+3.2
30 docs:	0.2406	0.2391	-0.6
100 docs:	0.1030	0.1035	+0.5
200 docs:	0.0600	0.0598	-0.3
500 docs:	0.0303	0.0303	-0.0
1000 docs:	0.0159	0.0159	-0.0
R-Precision:	0.3753	0.3880	+3.4

Table 7: Precision-recall data for SDR portion of TREC data corresponding to Figure 11.

4.7 Discussion

This section has presented an approach to document retrieval that exploits ideas and methods of statistical machine translation, and fits into the source-channel framework. With the EM algorithm, the parameters of the distillation model can be trained in an unsupervised fashion from a collection of documents. Experiments on TREC datasets demonstrate that even these simple methods can yield substantial improvements over standard baseline methods such as the vector-space approach, but without explicit query expansion and term weighting schemes, which are inherent in the translation approach itself.

The view of a user's interaction with an information retrieval system as translation of an information need into a query is a natural one. Exactly this formulation is made in a recent overview of issues in information science presented to the theoretical computer science community [10]. In this section we have attempted to lay the groundwork for building practical IR systems that exploit this view, by demonstrating how simple statistical translation models can be built and used to advantage.

5 Conclusion

We have presented approaches to two central problems in information retrieval; both approaches draw heavily from ideas in information theory. The results have been promising in both cases, suggesting that a source-channel model is a powerful framework for considering problems in information retrieval. The main contribution of this thesis, at a high level, will be a theoretical and empirical validation of this claim. More concretely, this thesis will propose new, high-performance information-theoretic approaches to document retrieval and classification.

5.1 Directions for future work: Retrieval

More powerful distillation models

It is reasonable to expect that, just as IBM's more complex models of translation yielded more accurate translations, so should more complex models of the query generation process offer better retrieval performance. For example, one of the fundamental notions of statistical translation is the idea of a word *fertility*, which measures how many words in a target language are necessary to capture the meaning of a word in a source language. While there appears to be no good reason why a word selected from a document should generate more than a single query term per trial, we should allow for *infertility* probabilities: the probability that a word from a document doesn't participate in the distillation process. For instance, stopwords such as THE and OR are unlikely to give rise to any query terms. The experiments in Section 4.6 accounted for stopwords in the traditional way: construct (by hand) a list of stopwords, and disregard such words in processing the collection, but infertility parameters would be a more principled way to account for this phenomenon. In a somewhat complementary way, the use of a *null word* in the document for generating spurious or content-free terms in the query (consider, for example, a query $\mathbf{q}$ = "FIND ALL OF THE DOCUMENTS...") could be useful.

The use of *distortion probabilities* could also be important: since people read documents from the beginning, words early in a document should probably be accorded more weight than those at the end. An easy way to account for this is to discount words according to their distance from the beginning of the document in constructing the document language model $l(w \mid \mathbf{d})$.

Translingual IR

While the amount of online text has grown dramatically in recent years, this information is in many different languages, and so only a fraction is accessible to any one person. Recently, much attention in the IR community has focused on how best to handle a setting where queries and documents may be in different languages [20].

Two standard approaches are to translate the query into the document language S using a dictionary, and to use a statistically-derived correspondence between the languages S and T. It has been shown empirically [42] that the latter approach yields better retrieval performance, at least in part because the "hard and fast"

dictionary lookup approach doesn't properly account for polysemy and synonymy. Within the translation-inspired retrieval approach, one could exploit the full context of the document to disambiguate, resolve word senses, and obtain a better distillation from the document to the query.

Real-world data

The retrieval system described here requires, for its construction, a collection of document/query pairs $\{d_i, q_i\}$ where d_i is a document relevant to the query q_i. Lacking access to a sufficiently large corpus fitting this description, we synthesized one in a manner described in Section 4.5. Even when trained on just this synthetic dataset, the retrieval system exhibited impressive performance on a selection of real-world evaluation data. Obtaining a sufficiently large collection of "real-world" document/query pairs for training—automatically compiled from user interactions with a commercial search engine, for instance—would further substantiate the retrieval-as-translation concept.

Document summarization

One can think of the document-to-query distillation model as a component in the retrieval framework we have developed. But it may be a useful tool in itself for automating the process of summarizing documents. Given a database of documents with their summaries, one could apply the exact procedure described in Section 4.5 to construct a summarization (as opposed to distillation) model. Although such a system may generate descriptive and relevant document summaries, these summaries won't be in meaningful English, since the models assume context-independent word generation. Some additional modeling of the target language would be necessary to produce readable summaries.

A specific implementation of this idea would be a system that automatically generates titles from documents. There do exist large, machine-readable collections of title/document pairs, and an automatic title annotator could be useful in many applications, such as the "video on demand" system under development at Carnegie Mellon University [14].

5.2 Directions for future work: Classification

More sophisticated codes

It is our belief that coding theory has more to say about classification. For instance, a useful class of error-correcting codes for digital transmission is *erasure codes*, which are robust to some fraction of lost bits. If the PiCs produce probabilities, then one could view a classifier λ which is sufficiently indecisive ($\lambda(x) \approx 1/2$) as a "lost bit"; an ECOC classifier containing λ could ignore λ in attempting to recover the label of the document.

Although we presented evidence suggesting the benefits of random codes, there are settings in which one would expect a structured code to be preferable. For instance, performing a nearest-neighbor search in high dimensional space could be prohibitively expensive for high-throughput systems. However, one might still be able to reap the benefits of high-n error-correcting output coding without actually conducting the full search. Using a deterministic code with some structure, like a BCH code, may allow for replacing the $\Theta(nm)$ exhaustive search with a $\Theta(n)$ search at a slight cost in accuracy. Real-world digital encoding/decoding systems—such as modems, CD players, satellites, and digital cell phones—use linear codes, containing exponentially more codewords than the simple codes described here, and therefore must rely on structured codes.

Sparse data

The text collections used in Section 3.3 to evaluate the error-correcting output coding method all had the property that each label was well-represented in the data. This is not true of most real-world datasets, where some labels may appear only very infrequently. Exploring the application of ECOC to such datasets is an important extension of this work. Section 3.3 takes a baby step in this direction by considering a feature-based PiC, rather than Naive Bayes, which tends not to perform well on poorly-represented labels.

Documents with multiple labels

The theoretical arguments which argue in favor of random codes are predicated on the assumption, untenable in most real-world data, that the errors made by the individual predictors are uncorrelated. In fact, textual data often contain strong correlations, which a classifier ignores at its own peril. For instance, the *astronomy* and *space* classes in the *Yahoo* science category have a strong overlap in word usage—evidenced by the confusion matrices of classifiers we have constructed on this data. A promising direction for improvement is to combine the ECOC approach with some form of word or document clustering, by designing a code which captures the inherent "clumpiness" of the data. In particular, a well-engineered code could reflect a hierarchical decomposition of the problem: first determine if the document belongs to the supergroup (*astronomy*, *space*), and only then decide which of these classes is most appropriate.

Another promising approach for the multiple-label problem is to apply some type of distillation model, similar to the one described in Section 4, from a document to a set of labels. We have run some preliminary experiments applying Hidden Markov Models (HMMs) to multilabel classification, with encouraging results, and we hope to try to understand how this HMM approach fits into the distillation framework.

AI-related issues

The question of exactly why ECOC works so well, and why voting algorithms work in general, is an open one, and the subject of much scrutiny in the AI and statistics community [5]. A goal of this thesis work is to understand how ECOC fits into the developing family of voting algorithms, and why ECOC, and voting methods in general, work so well in practice.

5.3 Timetable

Summer, 1999	Conduct retrieval experiments on a corpus of "real-world" data Extend the distillation framework to handle translingual retrieval; conduct experiments on translingual dataset(s) Develop an information-theoretic framework for multilabel classification Conduct experiments on multilabel ECOC (OHSUMED, Reuters, Patents, CoRA, ...) Compare ECOC technique to other voting algorithms (e.g. bagging, boosting)
Fall, 1999	Pursue ideas in document summarization Participate in TREC-8 evaluation Compare multilabel ECOC to standard techniques (like kNN)
Winter 2000	Begin writing dissertation Write articles on retrieval (with results on real-world data) and ECOC classification (with multilabel generalization)
Spring 2000	Finish dissertation

References

[1] D. Aha and R. Bankert. Cloud classification using error-correcting output codes. *Artificial Intelligence Applications: Natural Resources, Agriculture, and Environmental Science*, 11:1:13–28, 1997.

[2] L. Bahl, F. Jelinek, and R. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Inteliigence*, 5:2, 1983.

[3] D. Baker and A. McCallum. Distributional clustering for text classification. In *Proceedings of SIGIR*, 1998.

[4] G. Bakiri and T. Dietterich. Achieving high-accuracy text-to-speech with machine learning. *Data mining in speech synthesis*, 1999.

[5] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, (to appear), 1999.

[6] A. Berger, P. Brown, S. Della Pietra, V. Della Pietra, J. Lafferty, H. Printz, and L. Ures. The candide system for machine translation. In *Proceedings of the ARPA Conference on Human Language Technology*, 1994.

[7] A. Bookstein and D. Swanson. Probabilistic models for automatic indexing. *Journal of the American Society for Information Science*, 25:312–318, 1974.

[8] L. Breiman. Bagging predictors. *Machine Learning*, 26:2:123–140, 1996.

[9] L. Breiman. Bias, variance, and arcing classifiers. Technical report, Statistics Department, Stanford University TR-460, 1996.

[10] A. Broder and M. Henzinger. Information retrieval on the web: Tools and algorithmic issues. In *Foundations of Computer Science (FOCS) invited tutorial*, 1998.

[11] P. Brown, J. Cocke, S. Della Pietra, V. Della Pietra, F. Jelinek, J. Lafferty, R. Mercer, and P. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85, 1990.

[12] P. Brown, S. Della Pietra, V. Della Pietra, M. Goldsmith, J. Hajic, R. Mercer, and S. Mohanty. But dictionaries are data too. In *Proceedings of the ARPA Human Language Technology Workshop*, 1993.

[13] P. Brown, S. Della Pietra, V. Della Pietra, and R. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993.

[14] M. Christel, T. Kanade, M. Mauldin, R. Reddy, M. Sirbu, S. Stevens, and H. Wactlar. Informedia digital video library. *Communications of the ACM*, 38(4):57–58, 1995.

[15] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the World Wide Web. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98)*, 1998.

[16] W. Croft and D. Harper. Using probabilistic models of document retrieval without relevance information. *Journal of Documentation*, 35:285–295, 1979.

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

[18] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[19] J. Gemmell. ECRSM – erasure correcting scalable reliable multicast. Technical report, Microsoft (TR 97-20), 1997.

[20] G. Grefenstette, editor. *SIGIR Workshop on cross-linguistic information retrieval*, Zurich, Switzerland, 1996.

[21] G. James. *Majority vote classifiers: theory and applications*. PhD thesis, Stanford University, 1998.

[22] G. James and T. Hastie. The error coding method and PiCTs. *Journal of Computational and Graphical Statistics*, 7:3:377–387, 1997.

[23] F. Jelinek. *Statistical methods in speech recognition*. MIT Press, 1998.

[24] E. Kong and T. Dietterich. Error-correcting output coding corrects bias and variance. In *Proceedings of the 12th International Conference on Machine Learning*, pages 313–321, 1995.

[25] K. Lang. Newsweeder: Learning to filter news. In *Proceedings of the 12th International Conference on Machine Learning*, pages 331–339, 1995.

[26] D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In *Proceedings of the European Conference on Machine Learning*, 1998.

[27] F. MacWilliams and N. Sloane. *The theory of error-correcting codes*. North Holland: Amsterdam, The Netherlands, 1977.

[28] A. McCallum and K. Nigam. A comparison of event models for Naive Bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[29] M. Pazzani, J. Muramatsu, and D. Billsus. Syskill & Webert: Identifying interesting web sites. In *Proceedings of the National Conference on Artificial Intelligence*, 1996.

[30] M. Perrone. *Improving regression estimation: Averaging methods for variance reduction with extensions to general convex measure optimization*. PhD thesis, Brown University, 1993.

[31] J. Ponte. *A language modeling approach to information retrieval*. PhD thesis, University of Massachusetts at Amherst, 1998.

[32] J. Ponte and W. Croft. A language modeling approach to information retrieval. In *Proceedings of the ACM SIGIR*, pages 275–281, 1998.

[33] J. Quinlan. Combining instance-based and model-based learning. In *Proceedings of the International Conference on Machine Learning*. Morgan Kaufman, 1993.

[34] S. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27:129–146, 1976.

[35] S. Robertson, S. Walker, M. Hancock-Beaulieu, A. Gull, and M. Lau. Okapi at TREC. In *Proceedings of the first Text REtrieval Conference (TREC-1)*, 1992.

[36] D. Rumelhart, G. Hinton, and R. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

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

[38] Y. Singer and R. Schapire. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, (to appear), 1999.

[39] H. Turtle and W. Croft. Efficient probabilistic inference for text retrieval. In *Proceedings of RIAO 3*, 1991.

[40] W. Weaver. *Translation*. MIT Press, 1955.

[41] J. Xu. *Solving the word mismatch problem through automatic text analysis*. PhD thesis, University of Massachusetts at Amherst, 1997.

[42] Y. Yang, J. Carbonell, R. Brown, and R. Frederking. Translingual information retrieval: Learning from bilingual corpora. In *Artificial Intelligence Journal special issue: Best of IJCAI-97*, 1997.

[43] Y. Yang and C. Chute. An application of expert network to clinical classification and Medline indexing. In *Proceedings of the 18th Annual Symposium on Computer Applications in Medical Care (SCAMC'94)*, volume 18 (Symp.Suppl), pages 157–161, 1994.