Empirical Development of an Exponential Probabilistic Model for Text Retrieval

ABSTRACT

Much work in information retrieval has focused on using a model of documents and queries to derive retrieval algorithms. Models are a useful alternative to the heuristic development of retrieval algorithms because they make explicit the assumptions of the model developer, allowing the assumptions to be examined and refined independent of the particular retrieval algorithm. Unfortunately, in many cases these models are improved on by intuition, and their assumptions are not carefully tested against actual data. Thus, when a model based retrieval algorithm fails, it is not clear whether the fault is in the algorithm or the underlying model.

In this paper, we devise a generative document model via analysis of actual corpora and queries. Our thesis is that a model so developed will be more accurate, and thus more useful in retrieval, as well as other applications, than a heuristic model. We test this hypothesis by learning from the corpus the best document model within a naïve Bayesian framework, and investigating both how closely this learned model matches the data and how well it can be used to perform retrieval.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*retrieval models*

General Terms

Theory, Experimentation

Keywords

Information Retrieval, Formal Models, Machine Learning

1. INTRODUCTION

Information retrieval is about determining which documents are relevant to a user's information need. In early work on information retrieval, this determination was based on heuristic judgments [15] – for example, that documents containing the user's query terms are likely to be relevant – followed by heuristic tweaking of parameters (such as term weights) to make the system work. Subsequently, attempts were made to avoid, or at least make explicit, these heuristic judgments by developing models of queries and documents that could be used to deduce appropriate retrieval strategies. Examples of such models include the multinomial and 2-Poisson models. These are both *probabilistic models* that posit that relevant and irrelevant documents are generated from certain probability distributions, use these distributions to determine probabilities of relevance for each document, returning documents with high relevance probabilities (in accordance with the Probability Ranking Principle [21]).

While probabilistic models are intuitively appealing and easy to work with, relatively little attempt has been made to test the accuracy of their assumptions against actual data. Often a model is developed without reference to data; the only actual interaction with text is when testing a particular retrieval system based on the model (e.g., [6, 13, 20]). At that point it is difficult to decide whether unsatisfactory retrieval is due to the retrieval system or the underlying model.

We are interested in developing a generative model that, instead of directly improving retrieval, better describes text. Our hope is that improvements in a generative model will propogate through any model-based retrieval algorithm, yielding better performance in retrieval. This hope is not always fulfilled, as has been illustrated by the relatively unsuccessful attempts to allow for term dependencies in retrieval [10]. But by focusing on a few empirically inaccurate and easily correctable flaws in current models, we find we are able to improve our model's performance on certain retrieval tasks. Even when performance does not improve, generative models provide a useful approach to understanding text. They can be tested against actual corpora, generalized to other applications, and may even, in their characterization of text, suggest promising performance-improving heuristics.

We believe that the best way to improve on a generative model is to examine a corpus and modify the model to better match the actual statistical properties of text. We are of course unable to avoid all unfounded assumptions. For example, we restrict ourselves throughout the course of our analysis to naïve Bayesian models. However, we significantly relax some of the assumptions of standard probabilistic models and explore a much larger space of possible models.

We begin by discussing relevant work in model based retrieval. We then give a brief background of Bayesian machine learning and naïve Bayesian retrieval. In Section 4 we examine the well-known multinomial model, which assumes that terms in a document follow a multinomial distribution. This model diverges in basic ways from the actual statistics of text, and we highlight some of these ways. We then show in Section 6 how to identify, via examination of the corpus, a better model chosen from a class known as the *one-parameter exponential families*, which includes the multinomial model as one instance. We conclude with a discussion of how to use this improved model for retrieval and present experimental results.

2. RELATED WORK

While textual analysis is not currently a driving force behind the development of IR systems, early systems were often built by first studying the statistical properties of text. For example, Sparck Jones [9] suggested using the inverse document frequency for term weighting, now a common practice, based on textual analysis. At the time large test sets were not available, so the analysis was performed on a very small scale. Sparck Jones's analysis was done on three corpora of only a couple hundred documents each. Recently, people have revisited textual analysis on newer and larger data sets. For example, Grieff [5] suggested new improvements to tf.idf by studying 85,000 Associated Press articles from TREC 1. However, despite the fact that probabilistic models have explicit assumptions that should make textual analysis straightforward, this recent work has not been within a probabilistic framework.

There are a number of probabilistic models, such as a binomial model [4] and the 2-Poisson model [20] where the statistical properties of the text corpus are fundamental to the model. More recently, work has been done on language models [19] and other variants of the traditional probabilistic models [17]. While these models do attempt to minimize unfounded assumptions, the decision of how to build on the previous models is often not based on textual analysis. For example, Jin, Hauptmann and Zhai [8] suggest using the probability that a query would be the title of a document to rank documents, without actually investigating whether queries look anything like document titles.

In our work, we use textual analysis to understand and explore models within a naïve Bayesian framework. This framework has been well studied and Lewis gives a good overview of naïve Bayes retrieval and classification [14]. One common model within this framework is the multinomial model. While it is not popular for information retrieval, it has been explored for that use [11] and it is common in text classification [14]. The work we present here differs from these models because we do not restrict our model to a particular (e.g., multinomial) family of distributions, but rather derive the best family of distributions directly from the text. New approaches have been taken to derive the model *parameters* from the text [22],[1] but they do not attempt to derive the actual *model* from the text.

[1]from title, sounds like defining new model? If so, say so – I'm pretty sure they're not defining a new model, but I don't exactly understand the point you're raising.

Our approach uses relevance judgments on past queries in a principled way to devise a retrieval strategy for future queries, and thus serves as an interesting instance of cross query learning. Similarly, Fitzpatrick and Dent [3] use textually similar past queries for relevance feedback. In contrast to this work, rather than using topically related past queries, we use all of the past queries to teach us about the overall document model for relevant documents.

3. A BRIEF BAYESIAN TUTORIAL

Our approach is motivated by the standard Bayesian machine learning framework. In this section, we discuss that framework in order to lay the groundwork for our approach. Those familiar with Bayesian machine learning can safely skip this section. Due to space constraints, we focus on the "what" and give short shrift to the "why" of this approach; such material can be found in several sources [1, 7].

Our probabilistic approach begins with the assumption that relevant and irrelevant documents for a query are drawn from two distinct probability distributions. Formally, if we let $\mathbf{d}$ denote a random document drawn from a distribution, and let variable $r = 1$ if a document is relevant and 0 otherwise, then $\Pr(\mathbf{d} \mid r = 1)$ (resp. $\Pr(\mathbf{d} \mid r = 0)$) denotes the probability that a randomly generated relevant (resp. irrelevant) document turns out to be $\mathbf{d}$,

The Probability Ranking Principle [21] suggests that documents should be ranked for the user in order of their probabilities of relevance, $\Pr(r = 1 \mid \mathbf{d})$. A standard application of Bayes Law shows that this ranking is independent of $\Pr(r = 1)$ and monotonic in a *ranking value (RV)*

$$\frac{\Pr(\mathbf{d} \mid r = 0)}{\Pr(\mathbf{d} \mid r = 1)}.$$

In practice, while the probability distributions above are assumed to *exist*, they are not *known* at the time of retrieval. Bayesian machine learning provides a principled way to estimate these distributions based on data such as an input query or labeled documents. To estimate the desired distributions, we assume that the correct distribution is one member of some specific *family* of distributions and, based on the query-related information provided, we attempt to choose a plausible distribution from that family.

Two questions must be answered to use this approach: (i) what family of distributions is used (a modelling question) and (ii) which distribution to choose from the family given the data (a model-fitting question). As an example of what *not* to do, we might declare our relevant-document distribution to be a uniform distribution on the set of labelled relevant documents. This is an extreme example of *overfitting*: we get a distribution that perfectly explains the labelled documents we have seen, but assigns probability zero to any document we have not seen. Such a distribution is useless for predicting the relevance of other documents.

To avoid overfitting, the standard approach in machine learning is to use a *prior*. This is a probability distribution *over possibile probability distributions*. In other words, we label the possible probability distributions by some parameter θ, then specify a prior $\Pr(\theta)$. When presented with some labelled data D drawn from the unknown distribution, we choose as a plausible distribution the most likely value of θ, i.e. $\text{argmax}_\theta \Pr(\theta \mid D)$. This may be algorithmically challenging, but is well specified. The approx is referred to as *Maximum A Posteriori (MAP)* estimation.

Ignoring algorithmic issues, then, the only thing needed in a Bayesian machine learning approach to labelling documents is a prior over probability distributions of relevant documents and a prior over probablity distributions of irrelevant documents.

4. EXAMPLE: MULTINOMIAL MODEL

As a concrete example, the multinomial model [11, 14] assumes that the family of probability distributions is precisely the set of multinomial distributions over words – that is, that a document is generated by repeated independent sampling from a probability distribution of words in the corpus. Under such a model, the probability distribution for the number of times d_t that feature t appears in a document is

$$\Pr(d_t \mid r = x) = \binom{\ell}{d_t} (\theta_t^{r=x})^{d_t} (1 - \theta_t^{r=x})^{\ell - d_t}, \quad (1)$$

where ℓ represents the document length (note the distributions for distinct terms are not independent, as they must some to the specified docoument length). Within this family, a particular distribution is chosen for each feature t by setting the parameter $\theta_t^{r=x}$.

Given some labeled relevant documents, one can hypothesize that the "correct" multinomial distribution is one where the probability of each term is equal to its empirical probability of occurrence in the labeled documents. Unfortunately, this sets the probability of any unseen word to 0, which cause any new document with a new word to get probability 0, an example of overfitting. To avoid this, one uses a prior that (in addition to restricting to multinomial distributions) rules out 0-probabilities on terms, instead giving prior probability to distributions that give each term in the corpus some small chance of appearing (e.g., [18]).[2]

4.1 Evaluating the Multinomial Model

The multinomial model is very popular because of its simplicity. But it is worth investigating how good a model it is. We first clarify "goodness." Clearly, the quality of predictions made using the model (family of distributions) depends on how well the distribution selected from the model "matches" the actual (unknown) distribution. If the chosen distribution is very different from the target, predictions using it are unlikely to be accurate. A bad choice may arise from a bad learning algorithm, but a more fundamental barrier arises if the family contains no distributions similar to the real distributions—in that case, no learning algorithm can find a good one. The gap between true distributions and model distributions is known as *bias* and is one of the two sources of error in a learning algorithm (the other, *variance*, reflects flaws in the choice of a distribution from the model).

The standard way to measure the "fit" of a distribution to some data is to ask how likely that data was to be generated by the distribution. In this framework, the "best fit" distribution in the model is the one that assigns the maximum probability to the observed data; the higher the probability, the better the fit. This distribution is known as the *maximum likelihood (ML)* distribution. We can therefore evaluate the fit of a model to the data by finding the maximum likelihood distribution for the model.

Note that this approach makes no mention of the prior. This is because our goal here is to investigate the *existence* of a good distribution in the model; we do not aim to use the distribution for anything else.

For the multinomial model, it is well known that the maximum likelihood parameter setting, for θ_t, in the absence of a prior, is the average rate of appearance of the feature in the labeled data. This means that θ_t is the total number of occurrences of the term divided by the sum of document lengths.

4.2 Results of the Evaluation

In a preliminary study, we evaluated how well the multinomial distribution actually captured the statistical properties of a particular corpus: the TREC documents from CDs 1 and 2 of the series. We fit a multinomial model to these documents and then compared the fit model to reality. Since the multinomial distribution depends on the length of a document as well as its probability parameter, we focused on two sets of 10,000 and 25,000 documents of similar length, thereby eliminating any length effects. We found substantial mismatches in a number of basic properties.

Our overall goal was to determine how similar to multinomial distributions the observed distributions of terms in documents were. Unfortunately even if each term was drawn from a multinomial distribution, its *empirical* distribution might not look multinomial.

To get around this problem, we average the observed distributions of many "similar" terms. As discussed above, the maximum likelihood multinomial-model parameter setting for a term is the average rate of occurrence of the term in the data (ie, the fraction of terms that are the specified term). Under a multinomial distribution, the actual rate of occurrence of a term will be tightly concentrated around its expected number of occurrences. It follows that most of the terms appearing at a given rate in the corpus share roughly identical parameter values in the multinomial model. If we aggregate these terms, we should smooth out the noise of individual empirical term distributions and recover a distribution close to the true one. (We omit formal details due to lack of space.) For this reason we binned terms with similar occurrence rates, and compared their average distribution with what the multinomial model predicts.

Figure 1 shows the average term distribution for terms that empirically appeared approximately once every 100,000 words and the term distribution the multinomial model would expect for terms with the same rate of occurrence. Since the multinomial distribution is roughly exponential, it appears as a nearly straight line on a log scale. On the other hand, the tail for the empirical term distribution is much heavier. The probability of multiple occurrences of a term in a doc-

[2] This is so prevelant that it feels a little silly to cite...

ument is much higher than would be expected in a multinomial model, while the probability of a term occurring alone is empirically lower than would be expected. The multinomial model assumes that whether a term occurs again in a document independent of whether it has already occurred in that document, while clearly from Figure 1 this is an inaccurate assumption. This behavior, also called "burstiness", has been observed by other researches such as Katz [12] and Church and Gale [2].[3]

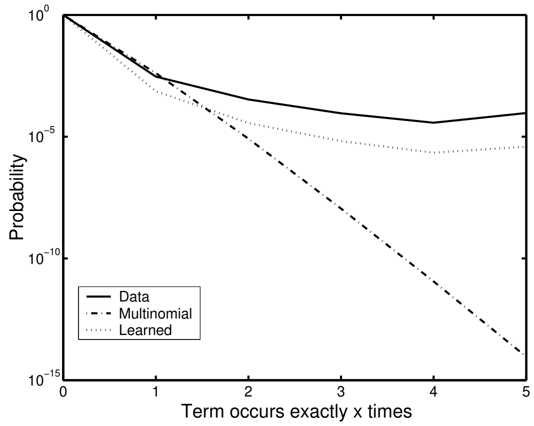

Figure 1: The empirical term distribution compared with what is expected for the multinomial model. Term occurrence appears to be more clustered in reality than the multinomial model expects.

5. CANDIDATES FOR A BETTER MODEL

Given the failure of the multinomial model to match the text, we sought a better model. However, we did want to preserve two important features of the multinomial model. First, computations over it, and in particular retreival algorithms using it, are very efficient. Second, its relatively "small" collection of distributions prevents overfitting (it is always possible to reduce the bias of a model by adding distributions; we wish to avoid that route). We therefore chose to focus on *naïve Bayesian models* [14] constructed from *One parameter exponential families.*

5.1 Naïve Bayesian Retrieval

While all probabilistic models assume that a document can be statistically described by certain features, and that the occurrence of those features is sufficient to determine whether the document is relevant or not, a naïve Bayesian model further assumes that document features are independent, given a relevance judgement. All interactions between features are ignored. This implies that the probability of a document is equal to the product of its feature probabilities and allows for efficient retrieval implementations. Many probabilistic IR models, can be expressed within the naïve Bayesian framework.

[3] While the empirical term distribution is not multinomial, the log of the term frequency distribution actually does look significantly more like a multinomial distribution. It appears almost straight on a log-log scale. Applying the machine-learning paradigm to this multinomially distributied quantity yields a variant of the standard vector space model, in which the "weight" of a term is set proportional to the log of its occurence count.

If document $\mathbf{d}$ has T features, with the tth feature represented as d_t, the probability of observing the document given a certain relevance judgment is thus

$$\Pr(\mathbf{d} \mid r = x) = \prod_{t=0}^{T} \Pr(d_t \mid r = x),$$

where $\Pr(d_t \mid r = x)$ denotes the probability of seeing d_t occurrences of feature t. While the independence assumption is obviously an oversimplification, it is common and useful because the number of features typically used to describe a document is very large. Having to account for all possible interactions between features would require an infeasible amount of time and space, as well as an infeasible amount of data to estimate those interactions.

While the feature set could be anything, in our analysis, we restrict the feature set to be words. Note that this same analysis could be done just as easily with whatever feature set we should choose to define, although the resulting statistical properties of that feature set would likely be different.

5.1.0.1 *Efficiency.*

By assuming that all features are independent, we can simplify our ranking value considerably.

$$\begin{aligned} RV &= \frac{\Pr(\mathbf{d} \mid r = 1)}{\Pr(\mathbf{d} \mid r = 0)} \\ &= \prod_{t=0}^{T} \frac{\Pr(d_t \mid r = 1)}{\Pr(d_t \mid r = 0)}. \end{aligned}$$

Taking the logarithm of this quantity (which helps avoid numerical instability caused by multiplying numerous small quantites) produces a sum of "term scores" that can be evaluated in much the same way as a traditional vector space model. Although the above shows a score per term in the corpus, as is observed elsewhere [20], it is possible to factor out the contributions of 0-occurrence terms to the probability and be left with a sum only over terms appearing in the document.

5.1.0.2 *Relation to the Multinomial Model.*

It should be noted that the multinomial model discussed above is does *not* fit our definition of a naive Bayesian model. Once the document length ℓ is fixed, a large number of occurrences of one term leaves fewer spaces for other terms to occur, so distinct term occurrence counts are not independent.

However, the closely related *Poisson model*, in which each term has an (independent) Poisson distribution (with the same parameters as in the multinomial model) does fit the Naive Bayes framework. Like other Na ıve Bayesian Models, the Poisson model does not produce documents of a fixed length, rather, it produces a distribution over document lengths.

5.1.0.3 *Coping with length.*

The fact that Naïve Bayesian models produce documents of varying lengths can have some undesirable consequences. In particular, the relevant and irrelevant document distributions may induce different document-length distributions, leading to questionable deductions such as "short documents are more likely to be relevant than long ones." While such a conclusion may be supportable, some prefer to apply "length normalization" to eliminate this dependence. From the Naive Bayesian model, one can derive in a principled fashion an "appropriate" and tractable length normalization scheme by conditioning the probability distributions on the desired length of a document. Because of the law of large numbers, the distribution of document lengths will be normal, and can be estimated by finding the mean and variance of the relevantand irrelevant documents. Thus each document can be normalized with respect to document length by multiplying it by this length factor. At the present time, we have not investigated this experimentally; instead, for the remainder of the paper, we shall simply ignore this length issue.[4]

5.2 Exponential Distributions

Having chosen the Naive Bayesian framework, we still need to define a family from which we can select (independent) probability distributions for the individual terms.

In an initial decision, we choose to limit to the possible distribution families to ones that could describe each term with a single parameter. This restriction ruled out many plausible distribution families, such as mixture models. However, restricting ourselves in this way reduced the risk of over-fitting, as there were less parameters to estimate with the same amount of data, and made for a simple and efficient model that translates easily into a vector space model.

Narrowing further, we chose the set of *one-parameter exponential families*. Exponential families are popular models in machine learning because they are quite general (the Poisson, Binomial, Uniform, and Gaussian families are all exponential families) but can still be optimized efficiently. Thus, much work has been done on how to best work with such families. For our application, they support efficient retrieval with the same complexity as standard models.

A one parameter exponential family takes the form

$$\Pr(d_t \mid \phi_t) = f(d_t)g(\phi_t)e^{\phi_t h(d_t)}, \tag{2}$$

where $g(\phi_t)$ is a normalizing constant equal to the inverse of $\int f(d_t)\exp\left(\phi_t h(d_t)\right) dd_t$. Recall that d_t is the number of occurrences of term t. The functions f and h define the specific exponential family. For example, for a binomial model, $f(d_t) = \binom{\ell}{d_t}$ and $h(d_t) = d_t$. For a Poisson model $f(d_t) = (d_t!)^{-1}$ and $h(d_t) = d_t$. A particular distribution is specified from the family by setting the parameter ϕ_t. We emphasize: the model is determined once by specifying f and h, which then become constants. Learning is performed by fitting the ϕ_t parameters.

6. HYPER-LEARNING A BETTER MODEL

Instead of imposing a particular exponential family, such as the Poisson family, on the term distributions, we propose that the "right" distribution family should be determined experimentally, prior to retrieval, directly from the data (the text, past queries and relevance judgments). This process, of examining the entire corpus to set the "hyper-parameters" f and h in our model, is similar in style to learning the parameters ϕ_t from labelled relevant documents; to distinguish this process from the process of learning the specific term distributions, necessary at query time, we call the preliminary process *hyper-learning*. Given a hyper-learned model, it is necessary at retrieval time to then learn the parameters that define the specific term distributions within the family as relate to the query (e.g., θ_t from Equation 1 in the case of the multinomial model). In this section we describe the hyper-learning process. By hyper-learning a better distribution family, we were able to create a new model that put increases the burstiness of term occurrences, and thus more accurately predicts the existence of our data.

6.1 Finding the Best Family

As was discussed above, a standard measure of "fit" between a given distribution and some data is the probability the distribution assigns to the data; within a distribution, the "best fit" distribution is the one that assigns Maximum Likelihood (ML) to the data. Formally, the quality of a particular model is equal to the maximum likelihood value we can achieve by choosing the specific term distributions by setting of ϕ_t parameters. If we define $Q(f, h, \phi_{\mathbf{t}})$ to be the likelihood assigned to the data by a particular collection of distributions from the model defined by f and h, then the best fit within the model is $\max_{\phi_{\mathbf{t}}} Q(f, h, \phi_{\mathbf{t}})$. If we want to find the best model, i.e. the one containing the best distribution for the data, then we wish to maximize the above quantity over choices of f and h. It is easy to see that we can conflate these two maximizations and find our best model by taking the values f and h from

$$\text{argmax}_{f,h,\phi_{\mathbf{t}}} Q(f, h, \phi_{\mathbf{t}}).$$

By setting $\phi_{\mathbf{t}}$ to be the maximum likelihood value we risk over-fitting. While this would not be a good setting for retrieval, it is acceptable because we are trying to find the model that best describes our data.

Let us first consider the problem of finding the best model to explain a particular set of documents, such as documents relevant to a particular query. Recall that the probability of observing a particular document can be expressed as $\prod_{t=0}^{T} \Pr(d_t \mid \phi_t)$. To find the probability of the entire document set we need to find the product of $\Pr(d_t \mid \phi_t)$ over all terms and over all documents. This can be expressed more succinctly because d_t is always a non-negative integer, and, in fact, is almost always zero. Therefor, without losing almost any information, we can only consider the cases where d_t is less than some small integer k. Grouping together for each term t the times it occurs i times, we can express the probability of observing the entire document set as

$$Q = \prod_{t=0}^{T} \prod_{i=0}^{k} \Pr(i \mid \phi_t)^{n_i^t},$$

where n_i^t refers to the number of times term t occurs i times. Because the probabilities of term occurrences are small, we work with the log of this value,

$$\log(Q) = \sum_{t=0}^{T} \sum_{i=0}^{k} n_i^t \log(\Pr(i \mid \phi_t)).$$

Finding f and h is simply a matter of finding the $2(k+1)$ values the two functions take at each possible i to maximize

[4]mention future work plan?

this value. In our experiments, we take k to be 5, as even a k of 2 correctly describes 99.9% of all term occurrences.

Our approach remains much the same when we want to find the best model to explain a *collection* of document sets. We aim to fix values f and h that explains *all* of these document sets. However, we naturally allow that each document set may reflect a different distribution drawn from the model. Thus, we use a *distinct* parameter ϕ_t^R for term t in *each* document set. Formally, if $\mathcal{R}$ represents the class of document sets, and $R \in \mathcal{R}$ is a particular document set, the objective function Q becomes

$$Q = \prod_{R \in \mathcal{R}} \prod_{\mathbf{d} \in R} \Pr(\mathbf{d} \mid \phi_t^R).$$

This equation can be mapped back to the "single document set" equation if we simply pretend that a given corpus term actually turns into a distinct term in each set. Thus, we focus our discussion on the notationally simpler single-set formula, knowing that it applies equally to studying a collection of document sets.

The problem of optimizing Q is a large multivariate optimization problem. There is no obvious analytic solution, so we turn to computational methods. As a proof of concept, we use a simple gradient ascent method. As we will see, Q is convex in the parameters ϕ_t but non-convex in the values f and h. Thus, we cannot guarantee that gradient ascent finds a globally optimum model. In the future, we hope to combine a more sophisticated non-convex optimization algorithm on the f and h values with the fast and simple convex optimization of the ϕ_t values.

For the moment, we take an iterative approach. Given some fixed f and h values, it is straightforward to find the (globally) optimum ϕ_t values. We then hold the ϕ_t values fixed and seek (locally) optimum f and h values given those ϕ_t values. We iterate this procedure until it converges.

6.1.1 The Best ϕ_t Given f and h

The probability of observing each document, and thus the objective Q, is concave with respect to each parameter ϕ_t. We can prove concavity by showing that the second derivative of $\log(Q)$ with respect to ϕ_t,

$$\frac{d^2 \log(Q)}{d{\phi_t}^2} = -N \sum_{i=0}^{k} h(i)^2 \Pr(i \mid \phi_t) + N \left(\sum_{i=0}^{k} h(i) \Pr(i \mid \phi_t) \right)^2,$$

is negative. Since $\sum_{i=0}^{k} h(i) \Pr(i \mid \phi_t)$ is equal to the expectation of $h(i)$, $E[h(i)]$, we can rewrite the second derivative as

$$= -N(E[h(i)^2] - E[h(i)]^2)$$
$$= -N\mathrm{Var}[h(i)].$$

Since variance is always positive, the second derivative with respect to ϕ_t is always negative, and $\log(Q)$ is concave with respect to ϕ_t.

Because $\log(Q)$ is concave with respect to ϕ_t, if we know the functions f and h, we can find the most likely ϕ_t for each term using convex optimization. To find ϕ_t we do a simple bisection search to find where first the derivative of $\log(Q)$ with respect to ϕ_t,

$$\frac{d \log(Q)}{d\phi_t} = \sum_{i=0}^{k} h(i) \left(n_i^t - N \Pr(i \mid \phi_t) \right),$$

is equal to zero.

6.1.2 The Best f and h Given ϕ_t

Unlike for ϕ_t, the objective function is *not* convex in f and h, as can be seen in Figure 2. Nonetheless, to find f and h we do a simple gradient ascent to a local optimum, using

$$\frac{d \log(Q)}{dh(i)} = \sum_{t=0}^{T} \phi_t \left(n_i^t - N \Pr(i \mid \phi_t) \right)$$

and

$$\frac{d \log(Q)}{df(i)} = \frac{1}{f(i)} \sum_{t=0}^{T} \left(n_i^t - N \Pr(i \mid \phi_t) \right),$$

the gradient functions, to aid our search.

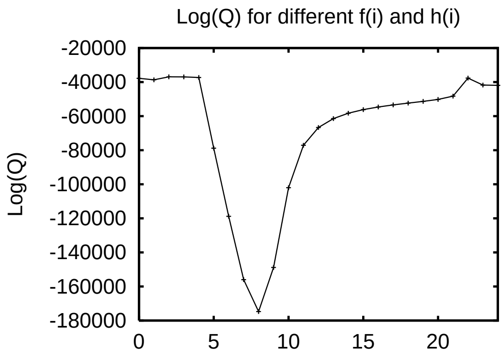

Figure 2: The probability of the data is not convex with respect to f and h. The graph plots the log-likelihood of the data under a convex combination of two distinct f and h values that achieved local maxima by optimizing from two different starting points.

Because $\log(Q)$ is not convex with respect to f and h, the starting point is very important. If we did not start somewhere good, we very well might converge to a poor local maximum. We began our hyper-learning with the Poisson family (exponential-family cousin of the multinomial family) as our starting point, setting $f(i) = (i!)^{-1}$ and $h(i) = i$.

6.1.3 Data Set

We hyper-learned the family of term distributions over sets of relevant documents. Each set was comprised of the relevant documents for a particular query from TREC 1 and 2 (queries 51-150). The query result sets ranged from containing just tens of documents and under a thousand unique terms, to containing almost a thousand documents and tens of thousands of unique terms.

d_t	f	h
0	1.1378	-1.1211
1	0.0307	2.1291
2	0.0049	3.1803
3	0.0015	3.6476
4	0.0007	3.9024
5	0.0014	4.0801

Table 1: The hyper-learned values for f and h.

We used sets of relevant documents for several reasons. First, as we will discuss in Section 7.1, we considered the problem of understanding of the family of distributions capturing relevant document sets more interesting for retrieval purposes. Additionally, because relevant document sets are considerably smaller than irrelevant document sets they also are considerably more varied within a given corpus. The largely overlapping irrelevant document sets would not allow us to generalize well. The smaller size of relevant document sets also made them more computationally feasible to work with.

Hyper-learning over all 100 document sets would have left us with no test data. Instead, we learned the best f and h for randomly selected "training" subset of up to 40 of the total 100 relevant document sets. This allowed us to test the learned model on the remaining "testing" document sets, and prove that our model would not overfit new queries. The functions f and h did not change drastically when hyper-learned on larger document sets (e.g., when increasing from a set of 30 to a set of 40), so it is likely that including all 100 would not have provided much additional information.

6.1.4 The Hyper-Learned Families

The functions f and h that we found as a result of hyper-learning over 40 relevant document sets can be seen in Table 1. The hyper-learned function f fell off much faster than its Poisson starting point, as can be seen in Figure 3. Because f is just a scalar in Equation 2, changes to its shape do not have a drastic effect on individual term distributions. However, f is the least term-dependent portion, and as such, its quick drop off probably reflects the quick drop off common to all terms between the probability of a term occurring zero times and a term occurring once or more.

The slope of the function h decreased considerably from its Poisson starting point, and its curve became concave, as can be seen in Figure 4. Because h is in the exponent of Equation 2, changes to it do have a significant effect on the resulting term distribution. The decreased slope and concavity probably reflect a de-emphasis on high term occurrences, correcting the Poisson model for the "burstiness" we observed.

Figure 1 shows an example distribution, illustrating that the new distribution family more closely matches our data. As expected from our textual analysis, we find a term distribution that has a heavier tail than predicted by the Poisson model.

In Table 2, we see that hyper-learning the distribution family does generalize well and make text data more likely than either the Poisson or multinomial models. To compute the

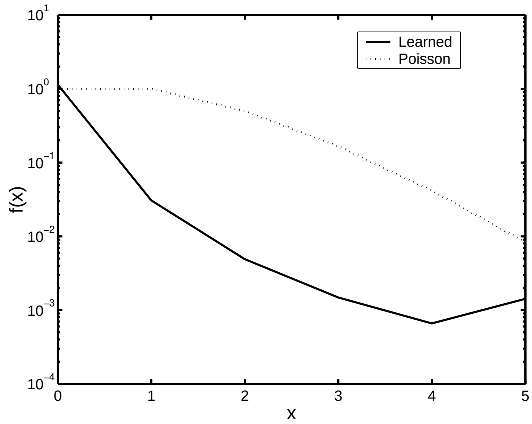

Figure 3: The hyper-learned function f, compared with the Poisson f.

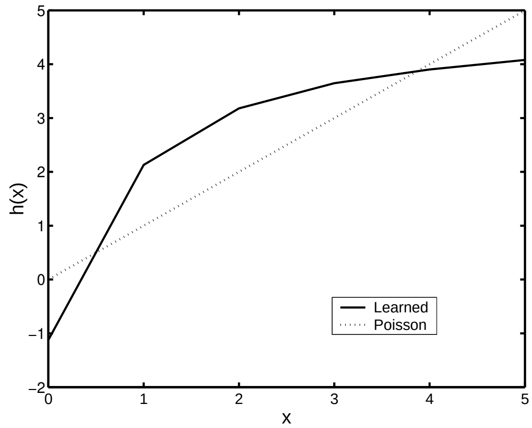

Figure 4: The hyper-learned function h, compared with the Poisson h.

	Multinomial	Poisson	Learned
average $\log(Q)$	-231,454	-231,368	-187,811

Table 2: The average log probability of observing one of the relevant document set the model wasn't trained on. The hyper-learned model generalizes well to better explain unobserved data.

values in Table 2, after we hyper-learned the best functions f and h over a set of 40 relevant document sets, we used that hyper-learned model to test the remaining 60 relevant document sets. Within those 60 remaining sets, we computed the maximum likelihood values for $\phi_{\mathbf{t}}$ from a subset of the documents, and then used the specific term distributions that that gave us to find the probability of observing the remaining documents. The new model clearly not only more effectively explains the data it was trained over, but generalizes well to the remaining documents.

7. RETRIEVING WITH THE NEW MODEL

We performed retrieval with the hyper-learned family of one parameter exponential distributions. We found that by improving on the Poisson generative model by hyper-learning from the text a better family of distributions, we were also able to improve on retrieval performance. In this section we first discuss how we estimated the model parameters ($\phi_{\mathbf{t}}$) to perform the queries and then we present our results.

7.1 Estimating the Distributions

To be able to rank a document we needed to specify both the relevant and irrelevant probability distributions for a term. We took a different approach to estimating the relevant and irrelevant distributions, and we discuss both processes separately here.

7.1.1 Relevant Distributions

In our estimation of the specific relevant term distributions (defined by $\phi_{\mathbf{t}}$) within our hyper-learned model, we assume we have observed some labeled information. This can either be a description of an information need (e.g., a keyword query), which we treat as a labeled relevant document, or actual labeled relevant documents.

For distributions in the one-parameter exponential family, all observations can be summarized, to the extent that they affect the parameter ϕ_t, with two *sufficient statistics*, τ_1 and τ_2 [1]. The statistic τ_1 represents the number of observations that have been made, and τ_2 represents the sum of all h values observed. Using this framework we can easily incorporate a prior by observing a certain number of "virtual documents" with the corpus statistics, and including these observations in our sufficient statistics.

The probability of ϕ_t based on the sufficient statistics is

$$\Pr(\phi_t) = [K(\tau_1, \tau_2)]^{-1} \left[\sum_{i=0}^{k} f(i) e^{\phi_t h(i)} \right]^{-\tau_1} e^{\phi_t \tau_2}, \qquad (3)$$

where $K(\tau_1, \tau_2)$ is the integral with respect to ϕ_t that normalizes the probability to sum to 1. This intregal is difficult to compute, but becomes irrelevant when we find the MAP estimate (most likely value) for ϕ_t by taking the derivative of Equation 3,

$$\frac{d\Pr(\phi_t)}{d\phi_t} = \Pr(\phi_t) \left(\tau_2 - \tau_1 \left(\frac{\sum_{i=0}^{k} f(i) h(i) e^{\phi_t h(i)}}{\sum_{i=0}^{k} f(i) e^{\phi_t h(i)}} \right) \right),$$

and set it equal to zero. It is zero when

$$\frac{\sum_{i=0}^{k} f(i) h(i) e^{\phi_t h(i)}}{\sum_{i=0}^{k} f(i) e^{\phi_t h(i)}} = \frac{\tau_2}{\tau_1}.$$

While there is also no obvious analytic solution for this problem, we can make a table of left hand side values as a function of ϕ_t. Once we observe our sufficient statistics, τ_1 and τ_2, we can use them to find the correct ϕ_t via table lookup.

Note that while we do not use it in our experiments, the Bayesian machine learning framework also provides a natural way to apply automatic relevance feedback. To do so, we could modify the framework to seek not just the MAP value of ϕ_t based on our initial observations about the relevant term distributions, but the joint MAP value of ϕ_t and a *relevance judgment for each unlabelled document*. The parameter estimate ϕ_t and the relevance judgments would feed off each other: a given ϕ_t would make documents that match the ϕ_t distribution look relevant, while labeling certain documents as relevant would push ϕ_t towards a value that generates those kinds of documents. For example, Nigam et al., implemented a multinomial model for classification that relied on only a few labeled documents, but used Expectation Maximization (EM) to include information from unlabeled documents [18]. We conjecture that our model, more accurate than the multinomial, would improve the performance of the Nigam et al. scheme.

7.1.2 Irrelevant Distributions

In order to find our ranking value, we also needed the term distributions within the irrelevant documents. We did not assume that we were provided with any labeled information about the irrelevant distributions. Instead, because most documents in a corpus are not relevant to any given query, we decided that it was close enough to assume that the feature distributions across the irrelevant documents were the same as the distributions across the entire corpus. This is not necessarily the case, particularly for the most interesting features in the relevant documents. For example, a very interesting relevant features such as those selected for a keyword query could possibly not occur in any irrelevant documents. However, our focus in hyper-learning distribution families was for query result sets. We focused less attention on understanding the distributions for the irrelevant documents. Additionally, assuming that all documents are irrelevant allowed us to estimate the irrelevant distributions before retrieval time, directly from the corpus.

The whole-corpus problem is in fact somewhat different. Unlike result sets for unknown future queries, the entire corpus is available in advance, and provides us with a large amount of information for estimating. It is therefore less important to have an entire *family* of distributions for the corpus and select the best actual distributions. Since we were *not* working with a family, we abandoned the restriction to an exponential family, and used the empirical distribution for each term in the corpus as our irrelevant document distribution.

7.2 Results

We first show a proof of concept that our model does improve retrieval when we can make a good estimate of the term distributions. We then discuss how the model can be used in a more typical environment, where the only information provided about the relevant distributions is a short query, and retrieval must be done in real time.

7.2.1 Using Labeled Documents

As proof of concept, we wanted to get a good idea of how our model worked when we actually had a good estimate of the term distributions. However, we also wanted to be sure it generalized. One way to do this is, similar to relevance feedback, to assume that there are a number of labeled relevant documents, use them to estimate the distributions, and then test over the remaining documents. This is the framework that matches most directly the standard machine learning model from Section 3. The occurrences pattern of a document feature in the labeled documents is used to MAP-estimate its probability distribution in relevant documents. In IR, the number of labeled documents used for estimating the relevant feature distributions ranges, and can be as many as half of the relevant documents [11].

We evaluated a model hyper-learned over 26 queries on the remaining 74 queries from TREC 1 and 2. We used 20 randomly selected relevant documents for our training, and assumed, as a prior, that we had observed 10 documents with the corpus distribution. Figure 5 shows the performance of the hyper-learned model compared with the Poisson model. The new model out-performs the Poisson model in all areas of the precision-recall curve. The average precision for the hyper-learned model (0.1643) is higher than that for the Poisson model (0.1411).

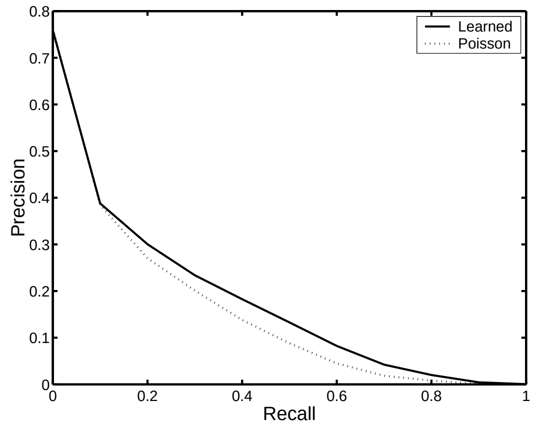

Figure 5: Precision-recall curves for retrieval using the hyper-learned model compared with Poisson model, using 20 training documents per query.

I also have a graph on varying the number of training documents, on a test run over five randomly selected documents (Figure 6). We can or cannot include it as space permits. I also could make one for varying the number of docs as a prior.

7.2.2 Using a Query

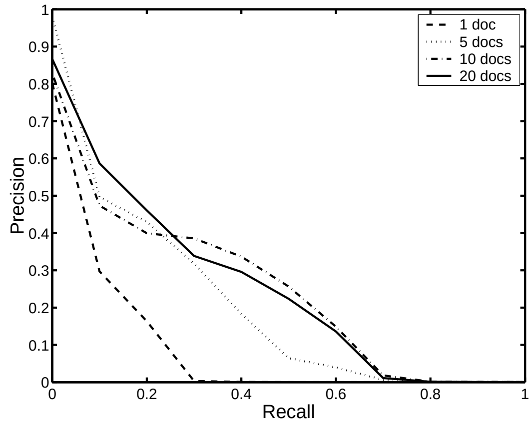

Figure 6: Precision-recall curves for retrieval using the hyper-learned model over five randomly selected queries not used in the hyper-learning. Shows affect of varying number of training documents.

A more common way for an information need to be specified is through a query. When the only information we have about the relevant term distributions is a query, we can rank documents using the following vector space type calculation:

$$RV = \sum_{t \in d} \left[\log\left(\frac{n_t^0}{n_t^{d_t}}\right) + \log\left(\frac{f(d_t)}{f(0)}\right) + (h(x) - h(0))\bar{h} \right] + \sum_{t \in q} \left[(h(d_t) - h(0)) \frac{h(q_t)}{c+1} \right].$$

The value c represents the number of documents we assume we've made with the corpus distribution distribution. The value for $\bar{h}$ is the average h for the corpus distribution times c plus one, since we have one new observation, the query. *Unclear. But I don't know if we should describe where this came from more or if it's a waste of space.*

This ranking value highlights the naïve Bayesian model's problem with length. All documents are given a default value regardless of whether the query term has occurred in it or not, and that value is affected by the number of terms in the document. In the case where we have a decent idea about the term distribution in relevant documents, this is often overpowered by the strength of the evidence in which terms occur or not. But in the case where we only know the query, it isn't, and retrieval performance is abismal.

This document value can be sidestepped if, instead, we assume that we don't learn anything about the terms that do *not* occur in the query. As can be seen in Figure 7, performance is comparable to tf.idf [16], slightly worse in areas of high precision, and better in areas of low precision (and always better than the Poisson model). By tweaking the way the query is incorporated more, we could probably get better performance. Because a query is not a document, it is not necessarily right to treat it as such. However, it would be good to come up with a good model for the query instead.

8. CONCLUSIONS AND FUTURE WORK

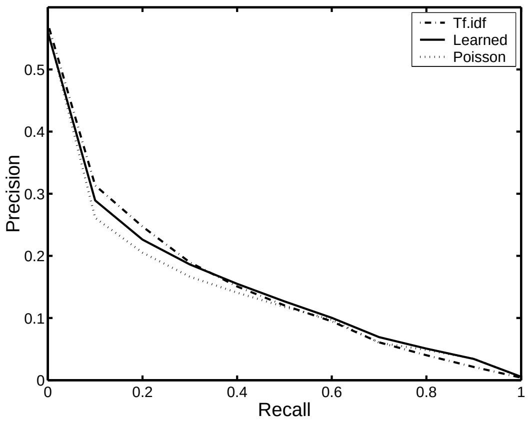

Figure 7: Precision-recall curves for retrieval using the hyper-learned model and Poisson model using only short queries. Compared with tf.idf.

By working with the statistical properties of text, we were able to learn a naïve Bayesian model that more closely matched the corpus than a multinomial or Poisson model. We discussed what was involved in learning such a model, and how we used it for retrieval. In learning more likely term distribution families, we were not only able to better describe our data, but we were also able to improve retrieval quality.

We used a simple gradient ascent method to hyper-learn the best distribution family for our model. There is still work to be done to find even better distribution families by hyper-learning global maxima for f and h. The best values would not be dependent on our starting point. Improving our search for the best distribution family will involve investigating some more sophisticated search functions, and arriving at an even better understand of our search space. Once we have good values for f and h learned from a variety of different text corpora, we would like to generalize them into analytic functions. This could allow us to make retrieval more efficient. We have also not yet tested normalizing for document length during retrieval, and intend to pursue this.

9. ACKNOWLEDGEMENTS

Removed for anonimity.

10. REFERENCES

[1] J. M. Bernardo and A. F. Smith. *Bayesian Theory.* John Wiley and Son Ltd, London, first edition, 1994.

[2] K. Church and W. Gale. Poisson mixtures. *Natural Language Engineering*, 1(2):163–190, 1995.

[3] L. Fitzpatrick and M. Dent. Automatic feedback using past queries: Social searching? In *20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1997.

[4] N. Fuhr. Probabilistic models in information retrieval. *The Computer Journal*, 35(3):243–255, 1992.

[5] W. R. Greiff. A theory of term weighting based on exploratory data analysis. In *Proceedings of SIGIR-98*, Melbourn, Australia, August 1998.

[6] A. Griffith, H. C. Luckhurst, and P. Willett. Using interdocument similarity information in document retrieval systems. *Journal of the American Society for Information Science*, 37:3 – 11, 1986.

[7] D. Heckerman. A tutorial on learning with Bayesian networks. Technical Report MSR_TR-95-06, Microsoft Research, Redmond, Washington, 1995. Revised June 1996.

[8] R. Jin, A. G. Hauptmann, and C. Zhai. Title language model for information retrieval. In *25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2002.

[9] K. S. Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21, 1972.

[10] K. S. Jones, S. Walker, and S. Robertson. A probabilistic model of information retrieval: development and status. Technical Report TR-446, Cambridge University Computer Laboratory, September 1998.

[11] T. Kalt. A new probabilistic model of text classification and retrieval. Technical Report IR-78, University of Massachusetts Center for Intelligent Information Retrieval, 1996.

[12] S. Katz. Distribution of content words and phrases in text and language modelling. *Natural Language Engineering*, 2(1):15–60, 1996.

[13] K. L. Kwok and M. Chan. Improving two-stage ad-hoc retrieval for short queries. In *Research and Development in Information Retrieval*, pages 250–256, 1998.

[14] D. D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In *EMCL*, pages 4–15, 1998.

[15] H. P. Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM Jouran of Research and Developement*, 1(4):309–317, 1957.

[16] K. McKeown, J. Klavans, V. Hatzivassiloglou, R. Barzilay, and E. Eskin. Towards multidocument summarization by reformulation: Progress and prospects. In *AAAI/IAAI*, 1999.

[17] K. Ng. A maximum likelihood ratio information retrieval model. In *Eighty Text REtrieval Conference (TREC-8)*, 1999.

[18] K. Nigam, A. K. McCallum, S. Thrun, and T. M. Mitchell. Learning to classify text from labeled and unlabeled documents. In *Proceedings of AAAI-98, 15th conference of the American Association for Artificial Intelligence*, pages 792–799, Madison, US, 1998. AAAI Press, Menlo Park, US.

[19] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Research and development in Information Retrieval*, pages 275–281, 1998.

[20] S. Robertson and S. Walker. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 232–241, Dublin, Ireland, 1994.

[21] C. J. vanRijsbergen. *Information Retrieval*. Butterworths, London, second edition, 1979.

[22] C. Zhai and J. Lafferty. Two-stage language models for information retrieval. In *25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2002.