Empirical Development of an Exponential Probabilistic Model for Text Retrieval

Jaime Teevan
MIT AI Lab
200 Technology Square
Boston, MA 02139
teevan@ai.mit.edu

David Karger
MIT LCS
200 Technology Square
Boston, MA 02139
karger@theory.lcs.mit.edu

ABSTRACT

Much work in information retrieval has been focused on the use of a *model* of documents and queries to derive retrieval algorithms based upon that model. 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 built 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

??

Keywords

??

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 [14] —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* which posit that relevant and irrelevant documents are generated from certain probability distributions, use these distributions to determine probabilities of relevance for each document, and return documents with high relevance probabilities (in accordance with the Probability Ranking Principle [20]).

While these 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 the text data is when testing a particular retrieval system (e.g., [6, 12, 18]). At that point it is difficult to decide whether unsatisfactory retrieval is due to the retrieval system or the underlying model.

Developing a discriminative model based on its performance in retrieval may be the fastest and easiest way to actually improve retrieval performance. However, we are interested in developing a generative model that, instead of directly improving retrieval, better describes textual data. We focus on generative models because they help illuminate clear paths for future improvement, can be generalized to other applications, and have foundations that can easily be tested against actual corpora. A better generative model does not necessarily lead to better performance, as has been illustrated by the relatively unsuccessful attempts to allow for term dependencies in retrieval [9]. But by focusing on the most empirically inaccurate and easily correctable flaws in current models, we find that in addition to the benefits of generative models described above, we are able to get improved retrieval performance.

We believe that the best way to improve on generative models is by examining a corpus, matching our model more closely to 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 are ablt to relax some of the assumptions of standard probabilistic models and explore a larger space of possible models.

We begin Section 5 by examining 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 the ways that it does. 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. 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." We conclude with a discussion of how to use this improved model for retrieval and demonstrate that with it we get improved performance.

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 [8] 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 used a probabilistic framework.

There are a number of probabilistic models, such as a binomial model [4] and the 2-Poisson model [19] where the statistical properties of the text corpus are fundamental to the model. More recently, work has been done on language models [17] and other variants of the traditional probabilistic models [15]. 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 [7] 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 [13]. One common model within this framework is the multinomial model. While it is not popular for information retrieval, it is common for text classification [13], and has been explored for text retrieval [10]. 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 [21], but they do not attempt to derive the actual model from the text.

In building our model, we use sets of relevant documents to learn about the statistical properties of relevant documents. Researchers have attempted to use past queries in an attempt to improve future retrieval performance. For example, 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 SKETCH OF BAYESIAN MACHINE LEARNING

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.

Our probabilistic approach begins with the assumption that relevant and irrelevant documents for a query are drawn from some probability distribution. We are interested in the different distributions that arise for relevant and irrelevant documents. 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 $\Pr(\mathbf{d} \mid r = 1)$ denotes the probability that a randomly generated relevant document turns out to be $\mathbf{d}$, while $\Pr(\mathbf{d} \mid r = 0)$ denotes the probability that a randomly generated irrelevant document turns out to be $\mathbf{d}$.

We would like to use the Probability Ranking Principle [20] to rank specific documents by their probabilities of relevance, $\Pr(r = 1 \mid \mathbf{d})$. By applying Bayes' Law, we can express this in terms of $\Pr(\mathbf{d} \mid r = x)$.

$$\begin{aligned}\Pr(r = 1 \mid \mathbf{d}) &= \frac{\Pr(\mathbf{d} \mid r = 1)\Pr(r = 1)}{\Pr(\mathbf{d})} \\ &= \frac{\Pr(r = 1)}{\Pr(r = 1) + \Pr(r = 0)\frac{\Pr(\mathbf{d}\mid r=0)}{\Pr(\mathbf{d}\mid r=1)}}.\end{aligned}$$

For most retrieval problems an ordinal ranking is sufficient and $\Pr(r = 0)$ and $\Pr(r = 1)$ are fixed and independent of the document we are ranking. Because the above quantity is monotonic with respect to $\Pr(\mathbf{d} \mid r = 1)/\Pr(\mathbf{d} \mid r = 0)$, we can use it to rank documents, and avoid having to estimate the probability a document is relevant or irrelevant. We will refer to $\Pr(\mathbf{d} \mid r = 1)/\Pr(\mathbf{d} \mid r = 0)$ as the "ranking value" (RV) of a document.

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 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 reasonable distribution from that family.

Selecting the appropriate family is difficult. Using too large a family leads to over-fitting. If a family encompasses all possible distributions, we can use the model to match the data exactly, but this does not generalize for unobserved data very well. Using too small a family faces the risk of not matching the data well enough.

As a concrete example, the multinomial model [10, 13] 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. 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. Similarly, one can choose a multinomial model for the irrelevant documents by using the empirical probabilities of terms over the irrelevant documents.

This example is an instance of *Maximum Likelihood (ML)* estimation. Maximum likelihood estimation selects, from among all distributions in the family, the one that maximizes the probability of the data (e.g., the labeled documents). Although it is not obvious, the empirical probability distribution mentioned above turns out to be the maximum likelihood estimator for the multinomial model.

This approach immediately runs into a problem: using the empirical distribution derived only from the labeled information means that any feature that does not appear in the labeled information is assigned probability of zero; this in turn means that any document containing that feature is assigned zero probability of relevance. This is a particular example of *over-fitting*—picking a probability distribution that looks great for the training data but does not generalize well.

To avoid over-fitting, one can apply various smoothing techniques that, for example, forbid zero probabilities. Bayesian machine learning takes a particular approach to this problem by defining a *prior* over the family of probability distributions. If we label the distributions by some parameter θ (so that we can refer to a particular document distribution as $\Pr(\mathbf{d} \mid \theta)$) then the prior is a probability distribution over θ. Instead of looking for the distribution that maximizes the likelihood of the data D, we look (again through a use of Bayes' law) for the θ that has maximum likelihood *conditioned on the observed data*, i.e., we select θ to maximize $\Pr(\theta \mid D)$. Selecting θ in this way is known as *Maximum A Posteriori (MAP)* estimation.

ML estimation can actually be viewed as a particular instance of MAP estimation. In the case of a uniform prior, where all values of θ are considered equally likely, we find using Bayes' Law that $\Pr(D \mid \theta)$ and $\Pr(\theta \mid D)$ are proportional. Maximizing the first (the ML estimate) also maximizes the second (the MAP estimate). But a prior gives us more flexibility—it can, for example, capture the intuition that a document distribution is unlikely to assign a zero probability to any feature. If the prior assigns some probability to such distributions then, even if the *data* contains no instances of the feature, the prior would prevent our selecting a distribution fitting this aspect of the data—in other words, it prevents over-fitting.

Of course, using a prior creates the added overhead of choosing one. There are standard techniques for doing so, however (true Bayesians argue that probabilistic methods that don't explicitly use a prior are nevertheless using one implicitly, so work is saved only at the expense of hiding one's assumptions). Indeed, any decision to restrict the model to a particular family of distributions is, in some sense, defining a prior in which all other distributions get probability of zero.

An alternative to the parameter estimation approach we describe above is to attempt to move directly from observed data to prediction, that is, to determine $\Pr(\mathbf{d} \mid D)$. This approach is in some sense even better founded— after all, there seems to be no reason to collapse the probability distribution over θ a single "best estimate" for the parameter. In general, however, computing $\Pr(\mathbf{d} \mid D)$ is done by integrating $\Pr(d \mid \theta)$ over the posterior distribution $\Pr(\theta \mid D)$. This calculation is often intractable, so MAP estimation is a widely used alternative. In our application, given the ultimate need for an efficient retrieval system, we have chosen to follow the MAP approach as well.

4. NAïVE BAYESIAN RETRIEVAL

While the Bayesian approach to retrieval can in theory be applied to any document probability distributions, many such distributions are intractable to work with in practice, either analytically or computationally. It is thus quite common to focus on *naïve Bayesian models* [13]. 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, such as the multinomial model, can be expressed within the naïve Bayesian framework.

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 both 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.

4.1 Efficiency

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

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

Because typical feature sets are large and sparse, such as words, most features do not occur in a majority of documents. Thus it is useful to find the ranking value only considering those features that actually do occur in a document.

$$RV = C \prod_{d_t \neq 0} \frac{\Pr(d_t = 0 \mid r=0)}{\Pr(d_t = 0 \mid r=1)} \frac{\Pr(d_t \mid r=1)}{\Pr(d_t \mid r=0)}, \qquad (1)$$

where $C = \prod_{t=0}^{T} \Pr(d_t = 0 \mid r=1) / \Pr(d_t = 0 \mid r=0)$ is a constant independent of the particular document. For the purposes of ranking, C can therefore be ignored. This is the basic probabilistic weighting function [19].

In practice, we could assume that for most features, those which are not content features, the probability of the feature appearing in a relevant document is equal to the probability that it appears in irrelevant documents. In this case the feature contributes nothing to the ranking value and can be ignored. We can use this approach to justify ignoring common features such as stop words or even to limiting our attention only to specific content features such as the words in the query.

The ranking value involves the product of many small probabilities, which may lead to numeric instability. For this reason, it is common to rank by the logarithm of the ranking value, which produces the same ordinal ranking. If we take the log of the ranking value from Equation 1, and assume only that the only content features are query terms, the ranking value turns into a sum, over query terms in each document, of certain "scores" for those query terms. This is simply a vector space model.

4.2 Dealing with length

Most naïve Bayesian models have a problem with length normalization. The models expect all documents to be the same length, determined by how many times the various terms are selected to appear. Document length can be inlcuded in a principled way by assuming that probability of observing a document is also conditioned upon the document length. This means the ranking value can be expressed as

$$RV = \frac{\Pr(\mathbf{d} \mid r=1, \ell)}{\Pr(\mathbf{d} \mid r=0, \ell)}$$

which, because we know the document actually is length ℓ, can also be expressed as

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

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 relevant and irrelevant documents. Thus each document can be normalized with respect to document length by multiplying it by this length factor.

5. EXAMPLE: MULTINOMIAL MODEL

One way to model the term distributions is to assume that they belong to the multinomial family, and can be expressed as

$$\Pr(d_t \mid r=x) = \binom{\ell}{d_t} (\theta_t^{r=x})^{d_t} (1-\theta_t^{r=x})^{\ell - d_t}, \qquad (2)$$

where ℓ represents the document length. Within this family, a particular distribution is chosen for each feature t by setting the parameter $\theta_t^{r=x}$. We evaluated this model and found that it failed to capture important aspects of the empirical distributions of features.

5.1 Fitting Model Parameters

To evaluate how closely the term distributions match multinomial distributions, we first needed to learn the appropriate parameters to best describe each term's particular distribution within the family of distributions (θ_t in Equation 2). As was discussed in Section 3, the Bayesian approach to learning this parameter would require the use of a prior. Our goal, however, is not to use the multinomial model for retrieval, but simply to ask how good a model it is. Thus, we choose not to use a prior. Instead, we applied maximum likelihood (ML) estimation to find the *best possible* distribution within the family for explaining the corpus. As was discussed in Section 3, ML estimation leads to over- fitting when used for retrieval; however, it does provide us with a good sense of the "most accurate possible" distribution for modeling the data. For the multinomial model, 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.

5.2 Compared to Text

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. Even if each term was drawn from a multinomial distribution, its *empirical* distribution might not look multinomial. Any particular term appears very rarely and does not really provide enough data to draw conclusions. However, should we gather a large number of terms drawn from the same underlying multinomial distribution, their "average" empirical distribution should look multinomial.

As discussed above, the maximum likelihood parameter setting for a term is the average rate of occurrence of the term in the data. Under a multinomial distribution, the actual rate of occurrence of a term will be tightly concentrated around its expected number of occurrences. Thus, if we assume the underlying term distribution is multinomial, we can be confident that by aggregating many terms that occur the same number of times overall, we are in fact aggregating many terms of roughly the same distribution. 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 4,400 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 document 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 [11] and Church and Gale [2].

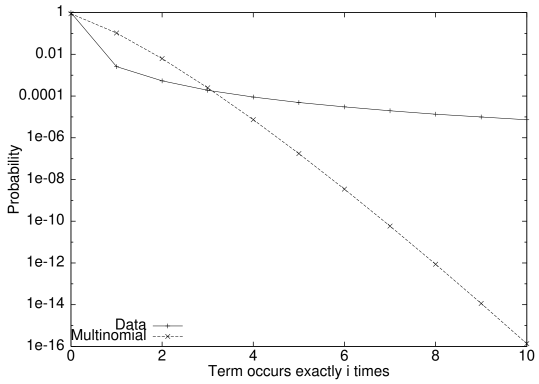

Figure 1: The empirical term distribution compared with what is expected for the multinomial model. Term occurrence appears to be more clustered in reality than the multinomial model expects.

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. This suggests choosing the *logarithm* of the number of occurrences from a multinomial distribution, rather than the actual number of occurrences. This is interesting because under a simple version of the multinomial model, where the relevant distribution is found by treating the query as a relevant document, the resulting document ranking value is a variant of tf.idf. The standard implementation of tf.idf actually uses the log of the document frequency in its calculations. That the log of the frequency turns out to look more multinomial supports this common ad-hoc use of the $\log(d_t)$ in tf.idf.

6. SEARCHING FOR A BETTER MODEL

Instead of imposing a particular distribution family, such as the multinomial family, on the term distributions, we propose that the distribution family should be learned, prior to retrieval, directly from the data (the text, past queries and relevance judgments). We call this 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 2 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 less emphasis on large term occurrences, and thus more accurately predicts the existence of our data.

6.1 Exponential Distributions

Through the hyper-learning process, we wanted to find a model that would match the actual term distributions as closely as possible. At the same time, we had several restrictions on the model. It needed to be general, and not restricted to the particular corpus it was learned over. It also needed to be something that could be optimized efficiently, as well as used efficiently for retrieval. For this reason, we choose to limit the set of distribution families under consideration.

To keep the model simple and efficient, 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.

We further restricted ourselves to one-parameter exponential families of distributions. Exponential families are popular models in machine learning because they are quite general (both the multinomial and Poisson families are 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

$$\Pr(d_t \mid \phi_t) = f(d_t)g(\phi_t)e^{\phi_t h(d_t)}, \tag{3}$$

where $g(\phi_t)$ is a normalizing constant equal to the inverse of $\int f(d_t)\exp(\phi_t h(d_t))\, dd_t$. 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 multinomial 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 $h(d_t) = d_t$. Once we hyper-learn f and h to define our model, we can perform retrieval by setting each term's parameter ϕ_t, thereby selecting a particular probability distribution it.

6.2 Finding the Best Family

The multinomial and Poisson models are specific exponential families. But there are many others, and it is conceivably that other families contain distributions that tend to be more like real term distributions. Our goal was to use the statistical properties of text documents to find values for f and h that yielded the best model. For this, we needed to define the notion of "best." A natural way to measure the quality of a model is to ask how well it can explain the data. From our probabilistic perspective, we considered data to be "well explained" by a model if the model contained term distributions that would assign the data a high probability.

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, our goal is to find the f and h that maximize the maximum over all $\phi_\mathbf{t}$ of $Q(f, h, \phi_\mathbf{t})$. 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

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

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. 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. If we just consider the cases where $d_t = 0$, $d_t = 1$ or $d_t = 2$, 99.9% of all term occurrences are covered. 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 this value. In our experiments, we take k to be 5.

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.

6.2.1 *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

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

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.

6.2.2 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 gradients functions, to aid our search.

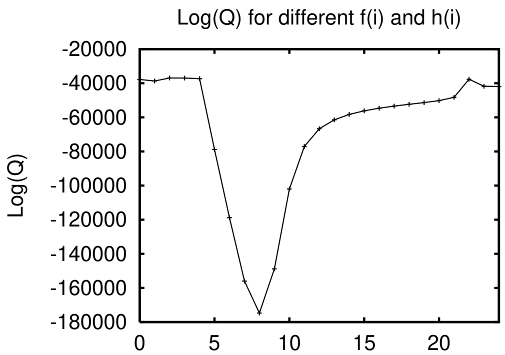

Figure 2: The probability of the data is not convex with respect to f and h. The two local maximum here are two different values of the most likely functions for f and h, found when optimizing from two different starting points.

Because $\log(Q)$ is not concave 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. However, if we had a good starting point, we would find better values. We began our hyper-learning with the Poisson family as our starting point, setting $f(i) = (i!)^{-1}$ and $h(i) = i$. We did this because the Poisson family of distributions is very similar to the multinomial family, while not being dependent on document length. By using this as our starting point, we knew that while we might not hyper-learn the globally best distribution family, we would find a better one.

6.2.3 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 were more interested in developing a good understanding of the relevant distribution family 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 the relevant document sets also made them more computationally feasible to work with.

Hyper-learning over all 100 document sets was prohibitively expensive. Instead, we learned the best f and h for randomly selected subset of up to 40 of the total 100 relevant document sets. This allowed us to test the learned model on the remaining document sets. The functions f and h did not change drastically when hyper-learned on larger document sets, so it is likely that including all 100 would not have provided much additional information.

6.2.4 The Hyper-Learned Families

The functions f and h that we found as a result of the 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 3, changes to its shape do not have a drastic effect on individual term distributions. But this quick drop off probably reflects the quick drop off 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 3, 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 5 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 values in Table 2, after 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

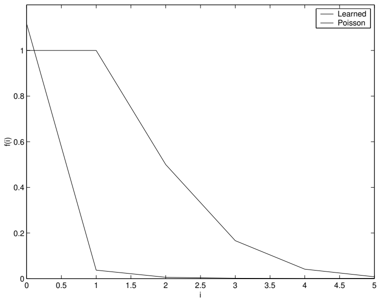

Figure 3: The hyper-learned function f, compared with the Poisson f. *Higher quality graph to come.*

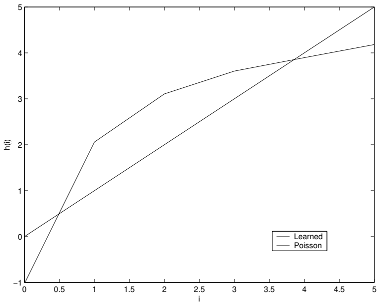

Figure 4: The hyper-learned function h, compared with the Poisson h. *Higher quality graph to come.*

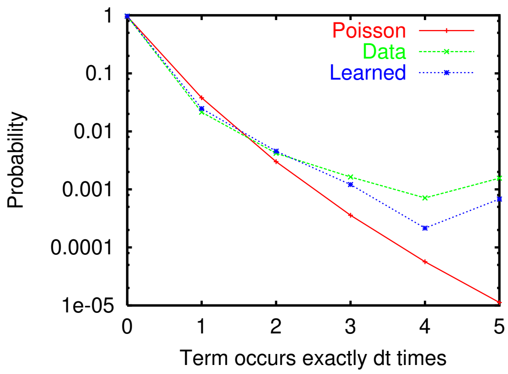

Figure 5: An example term distribution, as expected by the Poisson model and the learned model, compared with the corresponding empirical distribution. *Figure needs updating.*

	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 setw the model wasn't trained on. The hyper-learned model generalizes well to better explain unobserved data.

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 we were able to get better retrieval using the hyper-learned model than we were when using the Poisson model. 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

Recall from Equation 1 that the ranking value for a document is

$$RV \propto \prod_{t \in \mathbf{d}} \frac{\Pr(0 \mid r=0)}{\Pr(0 \mid r=1)} \frac{\Pr(d_t \mid r=1)}{\Pr(d_t \mid r=0)}.$$

To find the ranking value for a document we needed to specify both the relevant and irrelevant probability distribution for a term. Because the approach we took to estimating the specific term distribution differed based on whether we were estimating the relevant or irrelevant distribution, we discuss both processes separately here.

7.1.1 Relevant Distributions

There are a number of ways we could assume people inform the model of their information need. While the most common way is through a query, often it is assumed that the user has actually labeled some documents as relevant. 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. The number of labeled documents that people have used for estimating the relevant feature distributions can range from many, such as Kalt's experiments, where half of the relevant documents were labeled [10], to only a few.

In our estimation of the specific term distributions (defined by $\phi_{\mathbf{t}}$) within our hyper-learned model, we assume the presence of some labeled documents. For each query we use twenty labeled relevant documents, and test over the remaining documents. We use these documents to learn the parameters in the following way.

For distributions in the one-parameter exponential family, all observations can be summarized, to the extent that they affect the parameter, with two *sufficient statistics*, τ_1 and τ_2 [1]. The statistic τ_1 represents the number of all observations, and τ_2 represents the sum of all h values observed. The probability of ϕ_t based on the sufficient statistics is

$$\Pr(\phi_t) = [K(\tau_1, \tau_2)]^{-1} \left[\sum_{i=0}^{k} f(i) e^{\phi_t h(i)} \right]^{-\tau_1} e^{\phi_t \tau_2}, \qquad (4)$$

where $K(\tau_1, \tau_2)$ is the integral with respect to ϕ_t that normalizes the probability to sum to 1. This framework allows us to easily incorporate a prior by assuming we have observed a number of relevant documents with the corpus statistics. In our experiments we assume that we have observed 10 such documents. *Assuming there's time, we can also try different values for this, and see what works best.*

We can find the MAP estimate (most likely value) for ϕ_t by taking the derivative of Equation 4,

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

and setting that 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 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 relevance feedback. To do so, we could modify the framework to seek not just the MAP value of ϕ_t based on the labeled documents, 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 [16].

7.1.2 *Irrelevant Distributions*

In order to distinguish relevant documents from irrelevant, 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 documents. 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. While this is not necessarily the case, particularly for the most interesting features in the relevant documents, it allowed us to estimate the irrelevant distributions before retrieval time, directly from the corpus.

Our focus in hyper-learning distribution families was for query result sets. We focussed less attention on understanding the distributions for the irrelevant documents. The whole-corpus problem is in fact somewhat different. Unlike result sets for unknown future queries, the entire corpus is available in advance. 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 evaluated a model hyper-learned over 26 queries on the remaining 74 queries from TREC 1 and 2. The performance of the hyper-learned model, compared with the Poisson model, can be seen in Figure 6. The new model outperforms the Poisson model in all areas of the precision-recall curve. The average precision for the hyper-learned model (0.2750) is considerable higher than that for the Poisson model (0.1974). *Results currently over only 7 queries. Results for the remaining 67 queries to come shortly.*

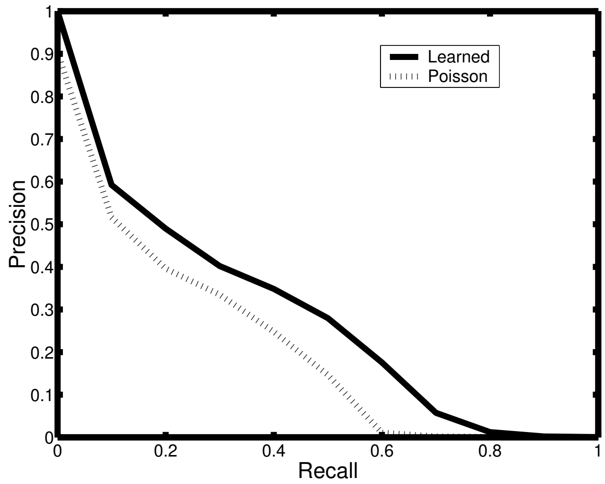

Figure 6: Precision-recall curves for retrieval using the hyper-learned model compared with Poisson model. ***(Currently only over seven queries.)***

8. CONCLUSIONS AND FUTURE WORK

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.

Because of how we developed our model, these results can easily be represented as a vector space model. If we treat the query as a labeled relevant document, the corresponding term weights can be seen in Table 3. The values assumes that the query term is observed only once – if it were observed more than once, the values would be different. Note that because using only a query we have so little labeled relevant information, we reduce the influence of our prior, and assume on one relevant document has been observed with

1	$1.6251h_t - 0.1519 + \log(n_0^t/n_1^t)$
2	$2.1507h_t + 1.3919 + \log(n_0^t/n_2^t)$
3	$2.3843h_t + 2.0553 + \log(n_0^t/n_3^t)$
4	$2.5118h_t + 2.3495 + \log(n_0^t/n_4^t)$
5+	$2.6006h_t + 3.9264 + \log(n_0^t/n_{5+}^t)$

Table 3: The corresponding term weights for a vector space model, assuming the query term occurs only once. The value for $\bar{h}$ is the average observed h_t for that term, $\sum_{d \in D} h(d_t)/n$.

the corpus statistics, instead of the 10 used in our experiments.

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

We would like to thank Stephen Robertson and Tommi Jaakola for their help with this work.

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] 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.

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

[9] 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.

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

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

[12] 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.

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

[14] 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.

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

[16] 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.

[17] 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.

[18] M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[19] 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.

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

[21] 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.