Empirical Development of an Exponential Probabilistic Model for Text Retrieval

Using Textual Analysis to Build a Better Model

Jaime Teevan
MIT AI Lab
Cambridge, MA 02139
teevan@ai.mit.edu

David R. Karger
MIT LCS
Cambridge, MA 02139
karger@theory.lcs.mit.edu

ABSTRACT

Much work in information retrieval focuses on using a model of documents and queries to derive retrieval algorithms. Model based development is a useful alternative to heuristic development because in a model the assumptions are explicit and can be examined and refined independent of the particular retrieval algorithm. We explore the explicit assumptions underlying the naïve Bayesian framework by performing computational analysis of actual corpora and queries to devise a generative document model that closely matches text. Our thesis is that a model so developed will be more accurate than existing models, and thus more useful in retrieval, as well as other applications. We test this by learning from a corpus the best document model. We find the learned model better predicts the existence of text data and has improved performance on certain IR tasks.

Categories and Subject Descriptors

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

General Terms

Experimentation

Keywords

Information Retrieval, Formal Models, Machine Learning

1. INTRODUCTION

The goal of information retrieval (IR) is to determine which documents are relevant to a user's information need. In early IR work, this determination was based on heuristic judgments [17] (e.g., that documents containing the user's query terms are likely to be relevant) followed by heuristic tweaking of parameters (e.g., 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. For example, probabilistic models are a common type of model used for IR. They posit that relevant and irrelevant documents are generated from probability distributions and use these distributions to determine probabilities of relevance for each document. They return, in accordance with the Probability Ranking Principle [23], documents with high relevance probabilities.

Even when model-based approaches fail to improve retrieval, they provide a useful approach to understanding text. The models can be tested against actual corpora, generalized to other applications, and may even, in their characterization of text, suggest promising performance-improving heuristics. However, 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 text data. Often a model is developed without reference to the data; the only actual interaction with text is when testing a particular retrieval system based on the model (e.g., [8, 15, 22]). At that point it is difficult to decide whether unsatisfactory retrieval is due to the retrieval system or the underlying model.

In this work our primary goal is to develop a generative probabilistic model that better describes text, and we rely on this better model to improve retrieval. Some previous work has tried to match underlying model assumptions closely to text through manual analysis of a small number of terms [2, 14]. In contrast, we match the assumptions to text computationally using a large text corpus (TREC). Thus, though we restrict ourselves throughout our analysis to naïve Bayesian models, we are able to significantly relax some of the assumptions of standard probabilistic models and computationally explore a much larger space of possible models.

Our hope is that improvements to the model will propagate 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 [12]. Still, 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 IR tasks.

We begin by discussing relevant work in model-based retrieval. We then give a quick overview of Bayesian machine learning and naïve Bayesian retrieval. We briefly examine the well-known multinomial model, which assumes that terms in a document follow a multinomial distribution, and show that this model unavoidably diverges from the actual statistics of text. We then discuss how we 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 the statistical properties of text corpora are fundamental to the use of probabilistic models, as well as to the use of other recent models [19, 21], the statistical properties have not necessarily been fundamental to the models' development, nor to understanding their assumptions. Most IR models do attempt to minimize unfounded assumptions, although often without understanding which ones actually are unfounded. For example, Jin, Hauptmann and Zhai [10] suggested using the probability a query would be the title of a document to rank documents, without investigating whether queries look anything like document titles.

Early IR systems did place focus on text during their development. For example, Sparck Jones [11] used analysis of three small corpora to suggest the use of inverse document frequency for term weighting, now a common practice, based on textual analysis. Recently, people have revisited textual analysis on newer and larger data sets. For example, Greiff [7] suggested improvements to tf.idf by studying 85,000 Associated Press articles from TREC. Because probabilistic models have explicit assumptions that make such textual analysis straightforward, some recent work has also been applied within a probabilistic framework. Church and Gale [2] and Katz [14] both look at empirical term distributions to build better models. The work presented here takes textual analysis a step further by, instead of imposing a model based on analysis, learning one computationally.

We derive our model within the well studied naïve Bayesian framework, of which Lewis provides a good overview [16]. Within this framework, researchers have considered a number of term distribution families to model text, with the selection sometimes based on textual analysis [2, 14]. Distributions that have been explored include binomial [5], multinomial [13, 16], Poisson [22], Poisson mixtures [2] and more [3]. The work we present here differs from this earlier naïve Bayesian work because we do not restrict our model to a particular family within the framework, but rather learn the best family from the text.

To learn the best distribution family, and thus our model, we do semi-parametric analysis of a single-parameter exponential family distribution, and there is substantial relevant statistical and machine learning literature on this [1, 6, 9]. Previously, attempts have been made to learn IR model *parameters* from text [24], but such work has not focused on learning the actual *model*.

In learning the model, we use relevance judgments on past queries in a principled way to devise a retrieval strategy for future queries. Thus our work serves as an instance of cross query learning. Similarly, Fitzpatrick and Dent [4] used textually similar past queries for relevance feedback. In contrast to their work, rather than using topically related past queries within a specific model, we use all past queries to teach us an 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. 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, 9].

We begin with the assumption that relevant and irrelevant documents for a query are drawn from two distinct probability distributions. Let $\mathbf{d}$ denote a random document, and variable r has the value one if a document is relevant and zero 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 [23] 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)*

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

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 modeling 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 could take our relevant-document distribution to be a uniform distribution on the set of labeled relevant documents. This would be an extreme example of over-fitting: the distribution would perfectly explain the labeled documents we had seen, but assign a zero probability to any document we had not seen. Such a distribution would be useless for predicting the relevance of other documents.

To avoid over-fitting, the standard approach in machine learning is to use a prior. A prior is a probability distribution over possible probability distributions. In other words, we label the possible probability distributions by some parameter θ and then specify a prior $\Pr(\theta)$. When presented with some labeled data D drawn from the unknown distribution, we choose as a plausible distribution the most likely value of θ, i.e. $\operatorname{argmax}_\theta \Pr(\theta \mid D)$. This standard approach is referred to as *Maximum A Posteriori (MAP)* estimation. As an example, nearly every probabilistic model assigns a prior that forces some nonzero probability onto every term in the corpus, even if the term does not appear in the labeled data.

Ignoring algorithmic issues, the only thing needed in a Bayesian machine learning approach to labeling documents is a prior over probability distributions of relevant and irrelevant documents and some information, such as labeled data, for selecting a distribution. While in practice it is standard to only address the model-fitting question (ii) through machine learning, we also address the modeling question (i).

4. EXAMPLE: MULTINOMIAL MODEL

As a concrete example, the multinomial model [13, 16] assumes that the family of probability distributions is precisely the set of multinomial distributions over words. A document is considered to be 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 term 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 is the document length. Within this family, choosing $\theta_t^{r=x}$ sets the particular distribution for each term.

The multinomial model is very simple, but not very good. The inadequacies of this model are well known, and can be found in the detailed analysis done by Church and Gale [2] and Katz [14]. However, their analysis was done over small sets of terms that were considered to have high content value, while we want the model we build to describe typical terms. For this reason, we briefly revisit the inadequacies of the multinomial model, looking at term distributions for hundreds of thousands of terms instead of just hundreds. Our analysis gives us an understanding of the corpus we are working with and a baseline to compare with as we improve on the multinomial model.

The quality of predictions made using a model (family of distributions) depends on how well the distribution selected from the model matches the actual unknown distribution. A bad match may arise from learning the distribution poorly, but a more fundamental barrier arises if the family contains no distributions similar to the real distribution. In that case, no learning algorithm could find a good match. 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 *maximum likelihood* distribution, which assigns the maximum probability to the observed data; the higher the probability, the better the fit. Because our goal in this case is to investigate the existence of a good distribution in the model, as opposed to using the distribution, we have no need to worry about over-fitting, and do not use a prior. For the multinomial model, in the absence of a prior, the maximum likelihood parameter setting for θ_t, is the average rate of appearance of term t in the labeled data.

We evaluated how well the multinomial family captured the statistical properties of a particular corpus, TREC 1 and 2. We fit a multinomial model to the corpus and then compared the fit model to reality. Our overall goal was to determine how similar to multinomial distributions the observed distributions of terms in documents were. Since a multinomial distribution depends on document length as well as its probability parameter, we focused on two sets of 10,000 and 25,000 documents of similar length. Even if a term were drawn from a multinomial distribution, its empirical distribution might not look multinomial. For this reason, we average observed distributions of many "similar" terms to smooth out noise. Because for multinomial distributions the actual rate of occurrence is tightly concentrated around its expected number of occurrence, we considered terms similar if they had similar rates of occurrence.

Figure 1 shows, on a log scale, the empirical term distributions found by averaging 500 terms that appear 15 times out of 3 million possible times in a corpus of 25,000 documents of similar length. This is compared with the maximum likelihood multinomial distribution for terms with the same rate of occurrence. The empirical distribution has a much heavier tail than the multinomial. That is, for the given average number of occurrences, the probability of multiple occurrences of a term in a document is much higher than a multinomial model would expect. The multinomial model assumes that whether a term occurs again in a document independent of whether it has already occurred, but clearly from Figure 1 this is an inaccurate assumption.[1] This behavior, also called "burstiness", that we found over a large number of terms in a large corpus is consistent with what others have observed [14, 2] through more in depth analysis on fewer "content" terms appearing in smaller corpora.

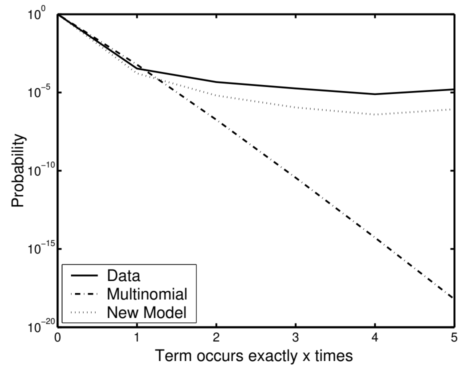

Figure 1: The empirical term distribution shows terms are more clustered compared to what would be expected for the multinomial model.

5. CANDIDATES FOR A BETTER MODEL

Given the failure of the multinomial model to match the text, we sought a better model. However, we wanted to preserve two important features of the multinomial model. First, computations over it, and in particular retrieval algorithms using it, are very efficient. Second, its relatively small collection of distributions prevents over-fitting. We therefore chose to focus on naïve Bayesian models [16] constructed from one parameter exponential families.

5.1 Naïve Bayesian Models

While all probabilistic models assume that a document can be statistically described by certain features (we assume terms), and that the occurrence of those features is sufficient to determine document relevance, a naïve Bayesian model further assumes document terms are independent given a relevance judgment. This is what makes naïve Bayesian models efficient for retrieval. Many probabilistic IR models fit within the naïve Bayesian framework.

Because the term probabilities are independent, interactions between terms are ignored. If document $\mathbf{d}$ has T terms,

[1] While the empirical term distribution is not multinomial, the distribution of its logarithm is roughly multinomial in shape. Applying the standard machine learning paradigm to this multinomially distributed quantity yields a variant of the standard vector space model, where the weight of a term is set proportional to the log of its occurrence count.

with the tth term represented as d_t, the probability of observing the document given a relevance judgment is

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

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

Efficiency

By assuming that all features are independent, we can simplify our ranking value (from Section 3) considerably.

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

Taking the logarithm of this quantity, which helps avoid numerical instability caused by multiplying numerous small quantities, produces a sum of per term scores. Although the above shows a score per term in the corpus, it is possible to factor out the contributions of zero-occurrence terms and be left with a sum only over terms appearing in the document [22]. This sum can be evaluated with much the same efficiency as a traditional vector space model (as can be seen, for example, in Section 7.2.2).

Relation to the Multinomial Model

It should be noted that the multinomial model discussed in Section 4 does does not fit our definition of a naïve 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 $\Pr(d_t) = e^{-\theta_t}\theta_t^{d_t}/d_t!$ does fit the naïve Bayes framework. Like other naïve Bayesian models, the Poisson model does not produce documents of a fixed length, rather, it produces a distribution over document lengths.

Coping with Length

That naïve Bayesian models produce documents of varying lengths can have undesirable consequences. 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, usually by ad hoc methods [3].

Length normalization can be pursued in a principled fashion. From the naïve Bayesian model, we could derive an appropriate and tractable length normalization scheme by conditioning the probability distributions on the given length of a document. Because in finding the probability of a document we sum many small independent quantities (term counts), the distribution of document lengths will be approximately Gaussian with computable mean and variance. This would allow us to normalize for length by multiplying each document's relevant and irrelevant probability by an easily computable length factor. At the present time, we have not investigated this approach experimentally; instead, for the remainder of the paper, we ignore length as an issue.

5.2 Exponential Families

Having chosen the naïve Bayesian framework, we still need to define a family from which we can select probability distributions for the individual terms. We choose to limit the possible distribution families to those that could describe each term with a single parameter. This limitation rules out many plausible distribution families, such as mixture models. However, this restriction reduces the risk of overfitting, as there are fewer parameters to estimate with the same amount of data, and makes 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 in machine learning because they are quite general (the Poisson, Binomial, Uniform, and Gaussian families are all exponential families) but can be optimized efficiently. Thus, much work has been done on how to best work with such families [1, 6]. 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 $\int f(d_t)\exp(\phi_t h(d_t))\,dd_t$. Recall that d_t is the number of occurrences of term t. The functions f and h, which apply in the same way to all terms, define the specific exponential family. A particular distribution for a particular term is specified from the family by setting the parameter ϕ_t. 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 $h(d_t) = d_t$. We emphasize: the model is specified once by choice of f and h; then, at retrieval time, 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 family should be determined experimentally, 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 labeled information; to distinguish it we call this preliminary process *hyper-learning*. Given a hyper-learned model, it is then necessary at retrieval time to learn the parameters that define the specific term distributions within the family as relate to the query (e.g., θ_t from Equation 1 for the multinomial model). In this section we describe the hyper-learning process. By hyper-learning a better distribution family over the TREC corpus, we found a new model that increased the burstiness of term occurrences and more accurately matched our data.

6.1 Finding the Best Family

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

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

By setting $\phi_{\mathbf{t}}$ to be the maximum likelihood value we risk over-fitting. While this would not be a good setting for $\phi_{\mathbf{t}}$ when performing retrieval, where we want to predict future data, it is acceptable in this case where we want to best describe our existing data, especially because, as we will discuss, there are very few f and h values.

Let us first consider how we would develop a model that accurately explains a single set of documents, such as those documents relevant to a particular query. Such a model is unrealistic, as the goal of the hyper-learned model is to accurately fit many queries, but is a good starting point. Recall that the probability of observing an individual 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 find the product $\Pr(d_t \mid \phi_t)$ over all terms and all documents. Because d_t is always a small non-negative integer, and, in fact, is almost always zero, we can, without losing almost any information, consider only the cases where d_t is less than some small integer k. Taking n_i^t to be the number of documents in which term t occurs i times, we can express the probability of observing the entire document set as

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

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 Q. In our experiments, we take k to be 5, but even a k of 2 covers describes 99.9% of all term occurrences.

However, our goal is to accurately fit many sets of documents, and not just one. To perform retrieval we would like to model all sets of documents relevant to any query. The approach remains much the same in this case. Our training data is a collection of document sets, and in hyper-learning we aim to fix values f and h that explain all of these document sets. Each document set may reflect a different distribution drawn from the model. Thus, we use a distinct parameter ϕ_t^R for same term t in each document set R. If $\mathcal{R}$ represents the class of document sets, and $R \in \mathcal{R}$ is a particular document set, the objective function Q becomes

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

Because a term can have a different distribution in each document set, we can treat the same term uniquely in each set. This allows us to map this equation back to the single document set equation. Thus, we focus our discussion on the notationally simpler single-set formula, knowing it applies equally to a collection of document sets.

The problem of optimizing Q is a large multivariate optimization problem. There is no obvious analytic solution, so we use computational methods, performing a simple gradient ascent. Given some fixed f and h values, it is straightforward to find (globally) optimum ϕ_t values. Similarly, given fixed ϕ_t values, we can find the (locally) optimum functions f and h. We iterate over these two procedures to find a maximum. Because Q is convex in the parameters ϕ_t but non-convex in the values f and h, we cannot guarantee gradient ascent finds a globally optimum model. A better optimum could be found by combining 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.

6.1.1 Finding 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 prove concavity by showing that the second derivative of $\log(Q)$ with respect to ϕ_t is negative. Note that because the probabilities of term occurrences are small, we sometimes work with $\log(Q)$ instead of Q. The second derivative of $\log(Q)$ is

$$\begin{aligned}\frac{d^2 \log(Q)}{d{\phi_t}^2} &= -N \sum_{i=0}^{k} h(i)^2 \Pr(i \mid \phi_t) \\ &\quad + N \left(\sum_{i=0}^{k} h(i) \Pr(i \mid \phi_t) \right)^2 .\end{aligned}$$

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

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

Variance is never negative, so the second derivative is never positive and $\log(Q)$ is concave with respect to ϕ_t.

It follows that 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 Finding the Best f and h Given ϕ_t

Unlike for ϕ_t, the objective function is not convex in f and h. 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.

Because $\log(Q)$ is not convex with respect to f and h, where we start our learning is very important. If we do not start somewhere good, we could converge to a poor local maximum. As our starting point, we used the Poisson

d_t	0	1	2	3	4	5
f	0.9121	0.0210	0.0034	0.0011	0.0005	0.0012
h	-1.4235	2.0466	3.0635	3.4882	3.6882	3.7714

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

family ($f(i) = (i!)^{-1}$ and $h(i) = i$). The Poisson family is commonly used to model term distributions and is the exponential-family cousin of the multinomial family.

6.2 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.

We used sets of relevant documents for several reasons. First, as we will discuss in Section 7.1, we consider the problem of understanding relevant document set distributions more interesting for retrieval purposes than understanding the corpus or irrelevant documents. Additionally, because the set of all documents relevant to a query is considerably smaller than all documents that are irrelevant to a query, relevant document sets are also considerably more varied within a given corpus than irrelevant document sets. The largely overlapping irrelevant document sets would not allow us to generalize well. Using the smaller relevant document sets also made our analysis more computationally feasible.

6.3 The Hyper-Learned Model

The functions f and h we found by hyper-learning over all 100 relevant document sets are shown in Table 1 and Figure 2. Both f and h in the hyper-learned model are substantially different from their Poisson starting points. The hyper-learned curve h is much flatter than the original Poisson function for nonzero x. This ensures that for any ϕ, there is less of a difference in probability of multiple occurrences, and thus a heavier tail (more burstiness), than for the Poisson model. Figure 1 illustrates this heavier tail on an example distribution for the maximum likelihood setting of ϕ, and suggests that the new distribution family does more closely match our data.

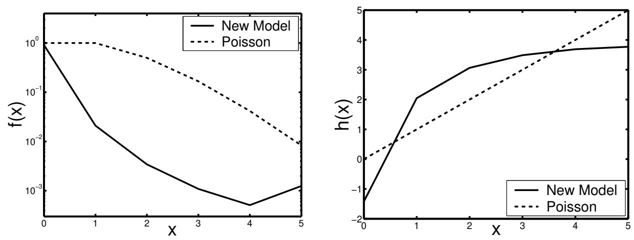

Figure 2: The functions f and h that define the hyper-learned family, compared with those that define the Poisson family.

To test how good our hyper-learned model is, we did not hyper-learn the best f and h over all 100 document sets, but instead hyper-learned on a randomly selected "training" subset of the 100 relevant document sets. The model we found when using fewer document sets was very similar to the one found using all 100 document sets, and by not learning over all of the sets we were able to test our model on the remaining "testing" sets. To test the model quality we used a family hyper-learned over 40 query result sets to predict the existence of the other 60 result sets. We computed the maximum likelihood $\phi_{\mathbf{t}}$ for each term from a subset of the documents in the testing subset and then used those specific term distributions to find the probability of the remaining documents. The average log probability of observing an unobserved relevant document set was much higher using the hyper-learned family (-428,164) than using the Poisson family (-517,171). In other words, the new model was less surprised than the Poisson model to see documents it had not seen before.

7. RETRIEVING WITH NEW MODEL

Given a better hyper-learned model, we turn to its use in retrieval, text classification, and relevance feedback tasks. We found that by improving the Poisson generative model by hyper-learning from the text a better family of distributions, we were also able to improve 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

Our analysis so far has focused on how well our model is able to explain labeled data. As was discussed earlier, in order use the model to generalize from labeled to unlabeled date (e.g., to assign relevance ranking values to unlabeled documents based on some relevance judgments) we must incorporate a prior so as to avoid over-fitting the labeled data. To be able to rank a document we needed to specify both the relevant and irrelevant probability distributions for a term, so we need priors for both. We took different approaches to the two distributions, which we discuss separately here.

7.1.1 Relevant Distributions

In estimating 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 single labeled relevant document, or actual labeled relevant documents.

One of the benefits of one-parameter exponentially families is that they have a so-called *conjugate prior* distribution that is extremely easy to work with. All information in the labeled data can be summarized, to the extent that it affects the choice of ϕ_t, with two *sufficient statistics*, τ_1 and τ_2 [1]. The statistic τ_1 represents the number of observations, and is the same for all terms. The statistic τ_2 represents the sum of all $h(d_t)$ values observed for a particular term. The prior is specified by an initial setting of τ_1 and τ_2, and can be thought of as reflecting the observation of a certain number of "imaginary" documents. The prior is updated by adding the true observations to the imaginary ones to get final values for τ_1 and τ_2.

Based on the sufficient statistics, the probability of ϕ_t is proportional to

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

This proportionality constant is difficult to compute for general ϕ, 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 $E[h(d_t)]$ is equal to the $h(d_t)$ values we observe empirically, namely 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, $\Pr(\phi)$ is convex, so we can optimize it by bisection search. In practice, for greater efficiency, we 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 a table lookup.

7.1.2 Irrelevant Distributions

To find our ranking value, we also need the term distributions within the irrelevant documents. Since for any given query, almost all documents are irrelevant, we chose not to apply our exponential-family-with-a-prior framework, and instead approximate the irrelevant document distribution of each term by its empirical distribution within the corpus. This whole-corpus problem is quite different. Unlike result sets for unknown future queries, the entire corpus is available in advance. Thus, we need not be concerned about over-fitting that will mismatch future unseen data.

Using the corpus distribution as the irrelevant distribution could potentially be inaccurate. A given query term, for example, might appear only in documents relevant to the query. However, it does simplify our approach, allowing us to set the irrelevant document distribution at preprocessing time, without a query. Furthermore, it is unclear that our exponential model, trained on the relatively coherent sets of relevant documents, would accurately model the much more diverse set of irrelevant documents, so there was no real incentive to assume the extra complexity.

7.2 Retrieval Results

In this section we show how our hyper-learned model performed on retrieval tasks. We first show that our model does improve retrieval when we make a good estimate of the term distributions. We then discuss how the model could 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

We first wanted to get a good idea of how our model worked when we actually had a good estimate of the term distributions, while being sure it generalized sufficiently. One way to do this, similar to relevance feedback, is to assume that there are a number of labeled relevant documents. These labeled documents we used to estimate the specific term distributions, and the distributions we used to find other relevant documents. This is the framework that matches most directly the standard machine learning model from Section 3, as the occurrences pattern of a document feature in the labeled documents is used to MAP-estimate its probability distribution in all relevant documents.

We evaluated a model hyper-learned (to find f and h) over 26 relevant document sets by testing it on the remaining 74 queries from TREC 1 and 2. To set our prior we imagined we had observed 10 documents with the corpus distribution. We then used 20 randomly selected relevant documents from each of the 74 test relevant document sets to learn the specific term distributions (ϕ_t) for each query. Figure 3 shows the performance on the remaining documents 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).

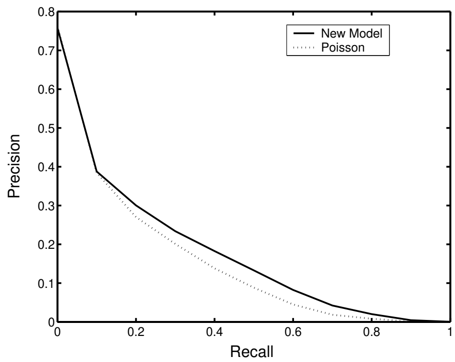

Figure 3: Results using the hyper-learned model, compared with results using a Poisson model. Twenty labeled documents were used to estimate the specific term distributions.

7.2.2 Using a Query

A query is a more common way to specify an information need. We treat a query as a single labeled relevant document and apply the classification scheme of the previous section. This gives us a RV of

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

where c represents the number of observations we've made for our prior, and $\bar{h}_t$ is our belief about term t assuming it doesn't appear in the query.

Our preliminary results using the titles of the TREC Topics 50-150 were poor. We speculate that this failure is related to the length normalization issue discussed earlier. As can be seen in the equation above, a document's ranking value has a component that is strongly affected by document length. Since a short query provides information about so few specific terms, this length effect dominates the presence or absence of specific terms. We took a quick, though less well founded, approach to correct for this by restricting our algorithm to only use those terms present in the query. This meant that for the document portion of the ranking value, we only summed over query terms. The results, displayed in Figure 4, show performance under this scheme is better in all areas of the curve than the Poisson model, as well as comparable to tf.idf [18], performing slightly worse in areas of high precision, and better in areas of low precision.

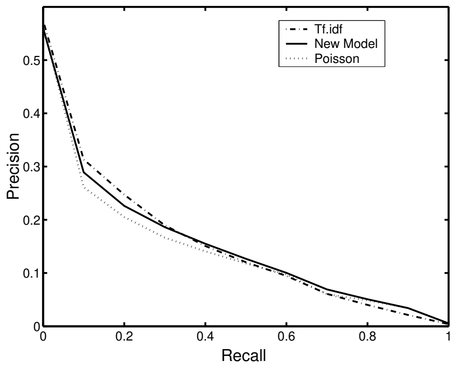

Figure 4: The hyper-learned model on short queries, compared with the Poisson model and tf.idf.

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 our corpus than multinomial or Poisson models. We discussed how to learn such a model and how to use it for retrieval. We found that 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 in some cases.

Although we used the hyper-learned model for information retrieval, an advantages of using a model is that it can be generalized to other applications such as classification and relevance feedback. For example, while we did not study it in our experiments, the Bayesian machine learning framework could be used for automatic relevance feedback, using unlabeled documents that seem relevant to modify its believed relevant-document distribution. Nigam et al. [20] got good results with this approach using the multinomial model, and we conjecture that our model, more accurate than the multinomial, could further improve performance.

We also believe that by learning a better model from the text, we could get even better performance gains. We currently use a simple gradient ascent method to hyper-learn the best distribution family, but would like to find a more global optima by incorporating smarter optimization techniques. Also, because it is likely that length has a significant impact on the quality of the model, it should be incorporated into the model in a principled way. Learning analytic functions for f and h could allow us to replace some of our table lookup steps by closed forms and enable us to handle terms that occur more than k times in a document.

It will also be interesting to investigate how general the model we learned over the TREC corpus is. If the same model could be learned from many different corpora, then we can apply our improvements to different situations, and even use the model to help us gain a better understaning of language. On the other hand, if the model varies greatly based on the text it describes, then learning the model becomes particularly important, making it possible to tailor retrieval to specific languages, specific domains, and even specific search preferences.

9. ACKNOWLEDGMENTS

The authors are grateful to Stephen Robertson and Tommi Jaakola for their input on this work.

10. REFERENCES

[1] J. M. Bernardo and A. F. M. Smith. *Bayesian Theory.* John Wiley & Sons, 1994.

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

[3] S. Eyheramendy, D. D. Lewis, and D. Madigan. On the naive Bayes model for text classification. In *Artificial Intelligence & Statistics*, 2003.

[4] L. Fitzpatrick and M. Dent. Automatic feedback using past queries: Social searching? In *SIGIR*, 1997.

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

[6] A. T. Gous. Adaptive estimation of distributions using exponential sub-families. *Journal of Computational and Graphical Statistics*, 7(3):388–396, 1998.

[7] W. R. Greiff. A theory of term weighting based on exploratory data analysis. In *SIGIR*, 1998.

[8] A. Griffith, H. C. Luckhurst, and P. Willett. Using interdocument similarity information in document retrieval systems. *JASIS*, 37:3–11, 1986.

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

[10] R. Jin, A. G. Hauptmann, and C. Zhai. Title language model for information retrieval. In *SIGIR*, 2002.

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

[12] 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, 1998.

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

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

[15] K. L. Kwok and M. Chan. Improving two-stage ad-hoc retrieval for short queries. In *SIGIR*, 1998.

[16] D. D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In *EMCL*, 1998.

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

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

[19] K. Ng. A maximum likelihood ratio information retrieval model. In *TREC-8*, 1999.

[20] K. Nigam, A. K. McCallum, S. Thrun, and T. M. Mitchell. Learning to classify text from labeled and unlabeled documents. In *AAAI*, 1998.

[21] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *SIGIR*, 1998.

[22] S. Robertson and S. Walker. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In *SIGIR*, 1994.

[23] C. J. vanRijsbergen. *Information Retrieval.* Butterworths, 1979.

[24] C. Zhai and J. Lafferty. Two-stage language models for information retrieval. In *SIGIR*, 2002.