

Probabilistic Modeling for Information Retrieval with Unsupervised Training Data

Ernest P. Chan⁽¹⁾, Santiago Garcia⁽²⁾ and Salim Roukos⁽³⁾

⁽¹⁾Credit Suisse First Boston
11 Madison Avenue
New York, NY 10010
ernest.chan@csfb.com

⁽²⁾Morgan Stanley & Co.Inc.
25 Cabot Sq.
London, U.K. E14 4QA
garcias@ms.com

⁽³⁾IBM T. J. Watson Research Center
POB 718
Yorktown Heights, NY 10598
roukos@watson.ibm.com

Abstract

We apply a well-known Bayesian probabilistic model to textual information retrieval: the classification of documents based on their relevance to a query. This model was previously used with supervised training data for a fixed query. When only noisy, unsupervised training data generated from a heuristic relevance-scoring formula are available, two crucial adaptations are needed: (1) severe smoothing of the models built on the training data; and (2) adding a prior probability to the models. We have shown that with these adaptations, the probabilistic model is able to improve the retrieval precision of the heuristic model. The experiment was performed using the TREC-5 corpus and queries, and the evaluation of the model was submitted as an official entry (ibms96b) to TREC-5.

1 Introduction

A generic task for machine learning is the automatic classification of a set of items into two classes. In the context of textual information retrieval (IR), the two classes may be the documents relevant/irrelevant to a specific topic. One common approach to such a classification task is to employ a Bayesian probabilistic model, of some parametric form, and estimate the parameters based on a set of training data that has been pre-classified. Such supervised training method is possible only when the definitions of the two classes are unvarying (e.g. the user is interested only in a single newspaper topic) and when training data is easy to obtain (e.g. articles from a sports magazine can be used to train a search engine for sports coverage in the World Wide Web). However, in many common applications of textual information retrieval, the user typically uses the search engine to search a new topic every time. Even if the same topic is desired each time, the user is often unwilling to spend effort in manually labeling the sample documents that are relevant to him/her.

In this paper, we report on an unsupervised algorithm to train a probabilistic model for information retrieval. This algorithm relies on a heuristic method to obtain the initial training data for the probabilistic model which can iteratively improve on the accuracy of classification. We

show how a well-known parametric probabilistic model in information retrieval, which hitherto has been used only when supervised training data is available, can be adapted to this unsupervised training method¹. The complexity of the adaptations needed reflects the common experience in IR that naïve probabilistic modeling by itself does not always produce good retrieval precision. We have applied this algorithm to the NIST TREC-5 task², and the results are submitted for the official TREC-5 evaluation [1]. The algorithm was shown to be effective in improving retrieval precision over the first-pass heuristic method, and is competitive with other top performers in TREC-5.

2 Bayesian modeling

2.1 The fundamental model

In this section we outline the Bayesian model commonly used in information retrieval. Since the model is well known, we refer the reader to Ref. 2 and 5 for details. We treat a document \mathbf{d} here as a bag of (unordered) n_d words. The only statistical feature that we use for classifying this document is the “term frequency” of each distinct word in the document, which is defined as the number of times this word occurs in the document. Hence we can regard \mathbf{d} as a vector of term frequencies of distinct words. Denote by κ the binary variable indicating whether the document \mathbf{d} is relevant to a specific query, such that $\kappa = 0$ when \mathbf{d} is irrelevant, and $\kappa = 1$ when relevant. Our goal is to estimate $p(\kappa|\mathbf{d})$, which is the probability of relevance of the document given its term frequencies. Following the usual rules of Bayesian statistics, we can write

$$p(\kappa|\mathbf{d}) = \frac{p(\kappa)p(\mathbf{d}|\kappa)}{p(\mathbf{d})}. \quad (1)$$

If we assume that the occurrence of a word is independent of the occurrence of other words, this can be shown to

¹ We use the term “unsupervised” to refer to our complete end-to-end IR system. If the probabilistic model is viewed in isolation, it is obviously “supervised”, since it relies on training data generated by the heuristic classifier.

² TREC stands for Text REtrieval Conference, an annual forum for the performance comparison of various text search systems on common corpora.

produce the same relevance ranking as

$$p(\kappa|\mathbf{d})=p(\kappa) \prod_{w \in V_d} \frac{p(d_w|\kappa=1) p(d_w|\kappa=0)}{p(d_w|\kappa=0) p(d_w|\kappa=0)}. \quad (2)$$

where the product is over only the words actually present in the document, denoted by the vocabulary V_d . Note that we have introduced the factor $p(d_w|\kappa)$, which is the probability of having term frequency d_w , and the factor $p(d_w=0|\kappa)$, which is the probability that the word w is absent in the document.

Now we have to decide on the precise form of the probabilistic function $p(d_w|\kappa)$. We have experimented with a variety of forms, including a maximum entropy model using the first few moments of the term frequency distributions of the training data as constraints, and some simple empirical histogram approaches with various binning and coarse-graining methods. However, we found that the best retrieval result is obtained with a parametric Poisson model, similar to the one described in [2]. The fact that a Poisson model is superior here is supportive of past empirical observations that word frequency distributions is approximately Poisson. In this Poisson model it is assumed that the rate of occurrence of a word in a document is proportional to the total document length n_d ,

$$p(d_w | \kappa, n_d) = p(d_w | \kappa, \alpha_w n_d) = \frac{e^{-\alpha_w n_d} (\alpha_w n_d)^{d_w}}{d_w} \quad (3)$$

Note that in addition to κ , the model is now conditioned upon n_d also, but this is a minor change from Eq.(1) – Eq.(2). The proportionality constant α_w is different for each word w , and is also a function of whether $\kappa=0$ or $\kappa=1$. It can be found using maximum likelihood estimation (MLE) based on the training data,

$$\begin{aligned} \hat{\alpha}_w &= \operatorname{argmax}_{\alpha_w} \prod_d p(d_w | \kappa, \alpha_w n_d) \\ &= \frac{\sum_d d_w}{\sum_d n_d}, \end{aligned} \quad (4)$$

where \mathbf{d} runs over all documents in the training sets for $\kappa=0$ or 1 respectively.

We have described above the Bayesian model, more specifically the Poisson model, previously used by various authors to model term frequency statistics for collections of relevant and irrelevant documents. This model has achieved some success in the context of supervised training. In order to apply this model to the unsupervised setting, and when the corpus of documents in question is very large, we have found that several adaptations are crucial to render it successful. We will describe them in the following sections.

2.2 Smoothing

For each query, there are on average only a small number of relevant documents, even for a large corpus like TREC. Hence the relevant models $p(dw|\kappa=1)$ are typically trained

on very limited amount of data. It is necessary to interpolate the relevant model with the irrelevant model $p(dw|\kappa=0)$ to avoid overfitting and to produce a more robust estimate of $p(\kappa|d)$ in Eq.(2).³ We use a linear interpolation commonly called “smoothing” in language modeling for speech recognition, such that $p(dw|\kappa=1)$ in Eq.(2) becomes $\lambda_w p(dw|\kappa=1) + (1-\lambda_w) p(dw|\kappa=0)$. (This change does not affect the parameter estimation in Eq.(5), since $p(dw|\kappa=1)$ and $p(dw|\kappa=0)$ continue to be trained exclusively on the $\kappa=1$ set and $\kappa=0$ set respectively.) Consequently, Eq.(2) becomes

$$p(\kappa | d) = p(\kappa) \cdot \prod_{w \in V_d} \left[\lambda_w \frac{p(d_w | \kappa = 1)}{p(d_w | \kappa = 0)} + (1 - \lambda_w) \right] \left/ \left[\lambda_w \frac{p(d_w = 0 | \kappa = 1)}{p(d_w = 0 | \kappa = 0)} + (1 - \lambda_w) \right] \right., \quad (5)$$

Note that λ_w is different for each word w , because for words that are present in many relevant documents, the estimation of $p(dw|\kappa=1)$ is more robust, and therefore λ_w should be larger for these words. Hence we have implemented a bucketing scheme, leading to three different values of λ_w . Denoting the number of relevant training documents that contain the word w by n_w , we have the three buckets of n_w ,

$$\begin{aligned} 0 < n_w \leq 5: & \quad \lambda_w = 0.005, \\ 5 < n_w \leq 20: & \quad \lambda_w = 0.02, \\ 20 < n_w : & \quad \lambda_w = 0.05. \end{aligned} \quad (6)$$

The buckets and their associated λ_w are estimated using the TREC-4 corpus to maximize retrieval precision there. Notice the small magnitude of λ_w , even for the highest bucket: this severe smoothing is necessary to obtain reasonable results with the probabilistic model.

2.3 Prior Probability

From Eq.(1) to Eq.(5), we have carried the prior relevance probability, $p(\kappa)$. In previous works with the Poisson model, this prior was always omitted as there was no reasonable way to estimate it. However, in our iterative method of training detailed in the following section, the documents initially receive a relevance score from a heuristic formula. Hence it is quite reasonable to utilize this relevance score as an estimate of the prior probability; in fact, we have found from our experiments that including this prior is crucial. The prior we use is simply the exponentiated initial score. The final relevance score we

³ Strictly speaking, we should interpolate with a model of a random document regardless of relevance. But such a model is very close to our irrelevant model, since such a random document is very likely to be irrelevant.

assign to a document is obtained by taking the logarithm of Eq.(5) together with the prior,

$$\text{score} \equiv \log p(\kappa | d)$$

$$= c \cdot \text{score}(\text{1st pass}) +$$

$$(1 - c) \cdot \sum_{w \in V_d} \log \left\{ \left[\lambda_w \frac{p(d_w | \kappa = 1)}{p(d_w | \kappa = 0)} + (1 - \lambda_w) \right] / \left[\lambda_w \frac{p(d_w = 0 | \kappa = 1)}{p(d_w = 0 | \kappa = 0)} + (1 - \lambda_w) \right] \right\}. \quad (7)$$

The mixing coefficient c between the prior and the posterior probability obviously depends on what heuristic first-pass scoring formula we use. It can be estimated by maximizing the retrieval accuracy based on the TREC-4 corpus. We find a value of $c=0.95$ to be reasonable for the Okapi first-pass scoring formula mentioned below.

3 Training the Model

We estimate the parameters α 's in Eq. (3) for the $\kappa=0$ and $\kappa=1$ sets of documents in an iterative way: first, all the documents in the corpus are scored according to their relevance based on a heuristic formula. Next, the documents with the top N scores are selected to form the $\kappa=1$ set, even though they are obviously not all relevant. The rest of the documents are regarded as the $\kappa=0$ set. The α 's are then estimated on these sets using Eq.(4), and the resulting model is used to rescore the documents. The hope is that the new top N documents based on the Poisson model relevance scores contain more truly relevant documents than those in the first-pass. This is indeed borne out by our experiments. In principle, one can iterate this process and obtain an ever-improving $\kappa=1$ training set, and thus an increasing retrieval precision. We will elaborate on these steps below.

To obtain the initial set of relevant articles for training, we rely on a well-known scoring formula called the Okapi [3] formula⁴. This formula assigns a relevance score to each document based on the word count⁵ statistics of the document and the query in a similar manner as the Poisson formula above, with the important difference that the parameters in the Okapi formula are all pre-fixed to some values found to be reasonable in general, and not tuned to the specific query in question. After the documents in our large corpus are thus scored, we choose the top 40 documents to be our $\kappa=1$ training set. For smaller corpus, one should pick fewer documents.

In principle, all the rest of the documents in the corpus will form the irrelevant set $\kappa=0$. However, when the corpus is very large, it is impractical and unnecessary to use so

many irrelevant documents for training. We can make use of the following observation in order to sample fewer documents: in Eq.(7), when $p(d_w|\kappa=1)$ is zero because no document in the $\kappa=1$ set contains the word w , there is no need to compute $p(d_w|\kappa=0)$ when $d_w > 0$ either. Hence for the purpose of estimating α of the irrelevant probabilities, we need only consider those words that are present in the $\kappa=1$ set. The only caveat is that we still need the probabilities $p(d_w=0|\kappa=0)$ even when w is not in the $\kappa=1$ set. Fortunately, these can be estimated much more easily than the full probability distribution $p(d_w|\kappa=0)$.

Based on this observation, for each word w present in the $\kappa=1$ set, we can sample some N_1 (I for *irrelevant*) w -containing documents from the $\kappa=0$ set in order to estimate $p(d_w|\kappa=0)$. (We choose $N_1=2000$ for the TREC corpora.) This sampling must be done such that the entire corpus is scouted uniformly for such w -containing documents. Note that this sampling actually results in a distribution $p(d_w|\kappa=0, d_w>0) = p(d_w|\kappa=0)/p(d_w>0|\kappa=0)$ conditioned upon the fact that the sample documents have $d_w > 0$. We must therefore adjust the resulting probability by multiplying the factor $p(d_w>0|\kappa=0) = 1 - p(d_w=0|\kappa=0)$ to obtain the true $p(d_w|\kappa=0)$.

For the estimation of $p(d_w=0|\kappa=0)$, we can count the number of documents in the entire $\kappa=0$ set which does *not* contain w , call it n'_w , and estimate $p(d_w=0|\kappa=0) \approx n'_w/N$, where N is the total number of documents in the corpus. (The number n'_w is often needed in the first-pass scoring such as the Okapi formula anyway.) This naïve estimation suffers from the problem that $p(d_w=0|\kappa=0)$ is actually highly dependent on the document length n_d (Longer documents should have more chance of containing w .) We choose to overlook this problem in this work.

We summarize in Table[1] the actual functions used in the estimations of $p(d_w|\kappa)$ under various circumstances. We will use the shorthand

$$\Psi(x | \alpha, n) \equiv \frac{e^{-\alpha n} (\alpha n)^x}{x},$$

$$p_0 \equiv n'_w / N,$$

$$V_{\kappa=1} = \text{set of all words in the } \kappa = 1 \text{ set.}$$

	$w \in V_{\kappa=1}$	$w \notin V_{\kappa=1}$
$d_w = 0, \kappa=1$	$\Psi(0 \alpha_w, n_d)$	1
$d_w > 0, \kappa=1$	$\Psi(d_w \alpha_w, n_d)$	0
$d_w = 0, \kappa=0$	p_0	p_0
$d_w > 0, \kappa=0$	$((1-p_0)/(1-\Psi(0 \alpha_w, n_d))) \cdot \Psi(d_w \alpha_w, n_d)$	(not needed)

Table 1: Exact form of $p(d_w|\kappa)$ for various κ and conditions on d_w and w .

4 The Experiment

We have performed retrieval experiments with our model described above on the official TREC-4 and 5 corpora for the “ad hoc” task, each consisting of about 500,000

⁴ The version of Okapi formula we used can be found in Ref. 4.

⁵ We incorporate also the “bigram”, or word-pair, counts in this first-pass scoring, as mentioned in Ref. 4.

documents. We use the 50 short official queries from each TREC. These short queries are typically only a couple of sentences long, and generally result in much lower retrieval precision than the long queries of previous TREC's. We apply various natural-language processing steps to all the documents and queries such as morphological analysis and stopword removal as described in Ref.4. The TREC-4 data are used for estimating some of the general parameters (parameters which do not depend on specific queries or words) such as the buckets for the smoothing constants λ_w in Eq. (7), the weight c in Eq.(8), etc.

Smoothing parameters	See Eq. (6)
Prior mixing coefficients c (Eq. 7)	0.95
No. documents in $\kappa=1$ set	40
No. documents sampled in $\kappa=0$ set ⁶	2000

Table 2: Values for general model parameters.

The values chosen for these and other parameters are summarized in Table [2]. The overall performance of the system is not very sensitive to these settings: a 10% change results in less than 1% change in average precision. Human relevance judgments for the queries are used for training these general parameters in TREC-4; however, they are used strictly for testing purpose in TREC-5. The TREC-5 evaluation is an official submission of IBM, code-named *ibms96b* in [1].

In performing the experiment, we have only iterated once: i.e. we used the Okapi formula to do the initial scoring, train the probabilistic model to rescore the documents, then stop.

The main criterion for judging the retrieval precision of a system is the average precision (AveP). Another useful

⁶ These 2,000 documents are chosen from the corpus *outside* the top 1000 scored documents from the previous iteration, instead of just outside the top 40 documents. This is a precaution to ensure that the documents we choose are indeed irrelevant.

	AveP	P5	P10	P30	P100
f	0.1557	0.4120	0.3320	0.2373	0.1560
u	0.1277	0.2800	0.2380	0.2087	0.1570
f+u	0.1637	0.3680	0.3200	0.2307	0.1722
f+u+b	0.1623	0.3960	0.3160	0.2413	0.1656
Knn	0.1585	0.3440	0.3240	0.2600	0.1726

Table 3: Results of experiments on TREC-5.

f: First-pass Okapi scores.

u: Poisson model for unigrams only.

f+u: Poisson model with prior, Eq.(5).

f+u+b: Poisson model with prior, added to non-probabilistic bigram scores. (Official submission *ibm96b*.)

Knn: Knn-rescoring. (Official submission *ibm96a*.)

criterion is the document level averages (P5, P10, P30, P100, etc.). The detailed explanations of these measures can be found in Appendix A of Ref. 1. Briefly, precision is the ratio of the number of *relevant* items retrieved to the total number of items retrieved. Average precision is the average of the precision value obtained after each relevant document is retrieved, and is usually regarded as the ultimate performance measure of a system. The document level average at cutoff= K (*abbr.* PK) is the precision value after K documents are retrieved. This measure is useful because often the user is only interested in the top few retrieved documents, so precision within these top documents are more important to him/her than the average precision over the entire set of retrieved documents.

In Table [3] below, we present our TREC-5 *ibms96b* evaluation results. We compare this result to several different models. First, we present the results obtained with just the first-pass Okapi scoring, which serves as a baseline (model f). Second, we show the results of our probabilistic model *without* adding the prior (first-pass) scores (model u). Third, we show the results when the prior is included (model f+u), which is the probabilistic model reported in this paper. Fourth, we present the results of *ibms96b*, which incorporates “bigrams” (*i.e.* pairs of words) as additional statistical features in the rescoring (model f+u+b). The scoring of bigrams (described in Ref. 4) is unrelated to the probabilistic model, and the bigram score is simply added linearly to the f+u score such that the bigram score has a weight of 0.95 versus a weight of 0.05 for the f+u score. The bigram result is included here merely to relate the pure probabilistic model f+u with the (slightly different) official submission f+u+b. Finally, we present the results of our other official model submitted to TREC-5 – *ibms96a*, which uses a different, non-probabilistic rescoring method called “Knn Rescoring” (model Knn, see Ref. 4).

We note that the probabilistic model without prior (u) is uniformly worse than the first-pass result (f). However, when we add the prior score (f+u), the average precision AveP is indeed higher than the first-pass average precision, indicating the effectiveness of the probabilistic model and the significance of a prior. In fact, this is (or rather, its close relative f+u+b is) the seventh best result among 32 official submissions to the TREC-5 ad-hoc automatic short-query category. (Extensions of this model continue to perform well in TREC-6, see Ref. 6.) Interestingly, some of the document level averages (P5 to P30) do not improve even when the prior is added. This is the same phenomenon with the Knn-rescoring, and is probably due to the small number of relevant documents present in the top40 training set (less than 24%, based on P30). If we run this same model on TREC-4, which has “easier” queries and thus larger number of relevant documents in the training set (about 35% or less), we see that both AveP and all the PK’s are improved⁷, as depicted in Table 4

	AveP	P5	P10	P30	P100
f	0.2014	0.5160	0.4440	0.3487	0.2356
f+u	0.2281	0.5200	0.4800	0.3787	0.2658

Table 4: TREC-4 results, for comparison.

5 Conclusion

We have shown in this paper how to construct a probabilistic model with unsupervised training data to classify documents based on their relevance to a query or topic. In particular, we show that in order for a probabilistic model to improve the classification accuracy over an initial heuristic classifier using the unsupervised, noisy training data produced by this initial classifier, two adaptations are needed: (1) severe smoothing of the relevant models with the irrelevant models; and (2) adding the probabilistic model score to a prior score given by the initial classifier.

One essential ingredient in any successful IR system is query expansion: the automatic inclusion of words not present in the original query but are nevertheless related to the topic of interest to the user. For example, we want to supplement the query “Tell me everything about Bill Clinton” with the words “U.S. President”, “White House”, etc. We have attempted to do this in an ad-hoc way in Ref. 4, with some success. A main advantage of using the probabilistic model described here is that it offers us a more principled way of achieving the same query expansion effect, and then to explore various adaptations and modifications of the fundamental model. For instance,

⁷ This is a somewhat unfair comparison, since the general parameters in Table 2 are trained on TREC-4 data, as we have discussed. However, we have found that even if we attempt to optimize the parameters on TREC-5, there is no great improvement in AveP and PK.

we can next consider how to incorporate conditional dependence between occurrence of two related words in the probabilistic framework. The main disadvantage is, however, that it is very computationally expensive to build the models, especially the irrelevant models. (It took 24 hours running six RS-6000 processors in parallel to collect term statistics for the irrelevant model.)

Recent research in IR has shown that building a truly superior retrieval system requires numerous specialized techniques and algorithms, many of which may appear to have little applications in solving more general problems in machine learning. We hope that by starting from a simple probabilistic framework, and specializing it only as the need arise, some insights gained here may find applications beyond IR.

Acknowledgments

This work is supported by NIST grant no. 70NANB5H1174. We thank Satya Dharanipragada for assistance in the pre-processing of the corpus and Robert T. Ward for stimulating discussions and various aspects of computing.

References

- [1] Harman, D. K. (ed.) 1997. *Proceedings of the Fifth Text Retrieval Conference (TREC-5)*. NIST Special Publication 500-238.
- [2] Robertson, S. E., Van Rijsbergen, C. J. and Porter, M. F. 1981. Probabilistic models of indexing and searching. In Oddy, R. N. et al. (Eds.) *Information Retrieval Research* (pp. 35-56). Butterworths, London.
- [3] Robertson, S. E., Walker, S., Sparck Jones, K., Hancock-Beaulieu, M.M., and Gatford, M. 1995. Okapi at TREC-3. In Harman, D. K. (ed.), *Proceedings of the Third Text Retrieval Conference (TREC-3)*. NIST Special Publication 500-225, 1995.
- [4] Chan, E. P., Garcia, S., and Roukos, S. 1997. TREC-5 Ad Hoc Retrieval Using K-Nearest-Neighbors Re-Scoring. In Harman, D. K. (ed.), *Proceedings of the Fifth Text Retrieval Conference (TREC-5)*. NIST Special Publication 500-238, 1997.
- [5] Robertson, S. E. and Walker, S. 1994. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In Croft, W. B. and Van Rijsbergen, C. J. (Eds.) *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, Dublin 1994*. (pp. 232-241). Springer-Verlag.
- [6] Franz, M. and Roukos, S., 1998. TREC-6 Ad Hoc Retrieval. In Harman, D.K. (ed.), *Proceedings of the Sixth Text Retrieval Conference (TREC-6)*. NIST Special Publication.