A General Language Model for Information Retrieval

Fei Song
Dept. of Computing and Info. Science
University of Guelph
Guelph, Ontario, Canada N1G 2W1
fsong@uoguelph.ca

W. Bruce Croft
Dept. of Computer Science
University of Massachusetts
Amherst, Massachusetts 01003
croft@cs.umass.edu

ABSTRACT

Statistical language modeling has been successfully used for speech recognition, part-of-speech tagging, and syntactic parsing. Recently, it has also been applied to information retrieval. According to this new paradigm, each document is viewed as a language sample, and a query as a generation process. The retrieved documents are ranked based on the probabilities of producing a query from the corresponding language models of these documents. In this paper, we will present a new language model for information retrieval, which is based on a range of data smoothing techniques, including the Good-Turing estimate, curve-fitting functions, and model combinations. Our model is conceptually simple and intuitive, and can be easily extended to incorporate probabilities of phrases such as word pairs and word triples. The experiments with the Wall Street Journal and TREC4 data sets showed that the performance of our model is comparable to that of INQUERY and better than that of another language model for information retrieval. In particular, word pairs are shown to be useful in improving the retrieval performance.

Keywords

Statistical language modeling, Good-Turing estimate, curve-fitting functions, model combinations.

1. INTRODUCTION

Information retrieval systems can be classified by the underlying conceptual models [3, 4]. Some of the commonly used models are the Boolean model, the vector-space model [12], probabilistic models (e.g., [11]), and the inference network model [3]. Recently, the statistical language modeling approach has also been applied to information retrieval. In a statistical language model, the key elements are the probabilities of word sequences, denoted as $P(w_1, w_2, \cdots, w_n)$ or $P(w_{1,n})$ for short. Such sequences can be phrases or sentences, and their probabilities can be estimated from a large corpus of documents. Statistical language modeling has been successfully used for speech recognition, part-of-speech tagging, and syntactic parsing [2].

However, estimating the probabilities of word sequences can be expensive, since sentences can be arbitrarily long and the size of a corpus needs to be very large. In practice, the statistical language model is often approximated by N-gram models.

Unigram: $P(w_{1,n}) = P(w_1)P(w_2)\cdots P(w_n)$

Bigram: $P(w_{1,n}) = P(w_1)P(w_2 \mid w_1)\cdots P(w_n \mid w_{n-1})$

Trigram: $P(w_{1,n}) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_{1,2})\cdots P(w_n \mid w_{n-2,n-1})$

The unigram model makes a strong assumption that each word occurs independently, and consequently, the probability of a word sequence becomes the product of the probabilities of the individual words. The bigram and trigram models take the local context into consideration: for a bigram, the probability of a new word depends on the probability of the previous word, while for a trigram, the probability of a new word depends on the probabilities of the previous two words.

Several people have applied statistical language models to information retrieval, including Ponte and Croft [10], Hiemstra [5], and the BBN group [6, 8]. Although the details differ between these approaches, the basic idea is the same. More specifically, we view each document as a language sample and estimate the probabilities of producing individual terms in a document. A query is treated as a generation process. Given a sequence of terms in a query, we compute the probabilities of generating these terms according to each document model. The multiplication of these probabilities is then used to rank the retrieved documents: the higher the generation probabilities, the more relevant the corresponding documents to the given query.

One obstacle in applying statistical language modeling to information retrieval is the sparse data problem, since a document is often small and its size and content are fixed. Our solution is to propose a new language model based on a range of data smoothing techniques, including the Good-Turing estimate, curve-fitting functions, and model combinations. Our model is conceptually simple and intuitive, and can be easily extended to incorporate probabilities of word pairs and word triples. The experiments with the Wall Street Journal and TREC4 data sets showed that the performance of our model is comparable to that of INQUERY and better than that of Ponte and Croft’s language model for information retrieval. In particular, word pairs are shown to be useful in improving the retrieval performance.

In the rest of the paper, we explain why the maximum likelihood estimate will not work for information retrieval due to the sparse data problem. After that, we describe in detail our new language model for information retrieval in section 3 and report our experimental results on two test collections in section 4. In section 5, we compare our model with the other language models for information retrieval. Finally, we conclude our paper and mention some of the future directions.

2. MAXIMUM LIKELIHOOD ESTIMATE

Given a document as a language sample, the question is how to estimate the probabilities of individual terms? A straightforward solution is to use the maximum likelihood estimate:

$$P_{mle}(t \mid d) = \frac{tf_{t,d}}{N_d}$$

where $tf_{t,d}$ is the number of occurrences of term t in document d, and N_d is the total number of term occurrences in document d.

Such a simple solution is unlikely to work due to the sparse data problem. It is common in statistical language modeling, but more serious in information retrieval for two important reasons. First, a document size is often too small. As a result, many terms would be missing in a document, which imply that their probabilities are zero. If such a term were used in a query, we would always get zero probability for the entire query, not helpful for ranking the documents. A solution to this problem is to allocate some probability mass to the missing terms so that their probabilities will always be greater than zero.

The other problem is that a document is fixed in size and content. Unlike a corpus, we can extend it with more documents if we want. A document is just a document: once it is written, its size and content are fixed. This makes it difficult to distinguish the effects of different missing terms in a document. As mentioned in [5], a document about information retrieval may have both terms “keyword” and “crocodile” missing, but for different reasons. When allocating probability mass to these terms, somehow we feel that the term “keyword” should have higher probability than the term “crocodile”, since “keyword” plays an important role in information retrieval. In other words, not all missing terms are equally likely but we cannot do much about them with the document itself.

The sparse data problem lead Ponte to explore the use of corpus data to stabilize a document model [9, 10]. More specifically, when a term does not appear in a document, the probability of the term in the corpus can be used as default. In cases where a term does appear in a document, an average probability is used to further smooth the document model. The average probability of a term is the sum of all the probabilities of the term across the documents in which the term occurs, divided by the number of documents in which the term occurs. After that, the probability of a term in a specific document is combined with the average probability of the term through a geometric distribution (called the risk factor). The intuition is that if a term occurs less frequent than the average probability in a document, we should make an adjustment to bring to it closer to the average. The advantage of this approach is that the risk factor is individualized for each term, but the disadvantage is that it is ad hoc and leaves little room for optimization. In addition, there is a potential problem with the default probabilities: some missing terms (e.g., words with the characteristics of a stopword) may be assigned higher values than the terms that actually appear in a document.

3. A NEW LANGUAGE MODEL FOR INFORMATION RETRIEVAL

Our solution is to propose a new language model for information retrieval based on a range of data smoothing techniques. First of all, we smooth each document model with the Good-Turing estimate, which is a discount model that allocates some probability mass to the missing terms. The current practice shows that the Good-Turing estimate often gives the best result among the available discount methods [7]; so it is also the method of our choice. Secondly, we expand each document model with the corpus model. The intention is to differentiate the contributions of different missing terms. For example, in a corpus about information retrieval, the term “keyword” will likely to happen more often than the term “crocodile”, so such information can be used to adjust the probabilities for the missing terms. Thirdly, we want to consider term pairs and expand the unigram model of a document with the bigram model. The intuition is that phrases such as term pairs would be useful in information retrieval, but unfortunately the existing research often did not show much improvement in the retrieval performance [1]. We would like to see that in the context of language modeling whether term pairs would bring any better results. Finally, there is the issue of combining different models. As we have just explained, we have the models for each document and the corpus, and further for each document, we have the unigram model and the bigram model. Instead of combining them in an ad hoc manner, we intentionally keep them separate and consider general ways of combining them.

3.1 Smoothing a Document Model with the Good-Turing Estimate

To address the sparse data problem, we need to allocate some probability mass to the missing terms. In the Good-Turing estimate [7], this is done by adjusting the raw tf scores:

$$tf^* = (tf + 1)\frac{E(N_{tf+1})}{E(N_{tf})}$$

Here, N_{tf} is the number of terms with frequency tf in a document, and $E(N_{tf})$ is the expected value of N_{tf}. Then, the probability of a term with frequency tf is defined as tf^*/N_d, where N_d is the total number of terms occurred in document d. In particular, when tf = 0, tf^* is reduced to $E(N_1)/E(N_0)$ and the probability mass allocated to a missing term becomes $E(N_1)/E(N_0)N_d$.

Since a document is fixed in size and content, obtaining $E(N_{tf})$ is almost impossible. In practice, we may hope to substitute the observed N_{tf} for $E(N_{tf})$. This creates two problems. For the terms with the highest frequency tf, their probabilities will be zero, since N_{tf+1} will always be zero. Furthermore, due to the small size of a document, the number of terms in some middle frequency levels may also be too small or even zero, resulting in an unstable or anomaly distribution.

One solution to these problems is to use a curve-fitting function to smooth the observed N_{tf}’s for the expected values. Table 1 shows a typical term distribution for a large corpus, taken from [7]. Obviously, a linear line will not fit the distribution properly. On the other hand, a simple geometric (exponential) curve will not fit the distribution either: Although N_{tf} decreases very quickly at the beginning, it slows drown considerably as tf gets much higher. After a number of experiments, we have developed a greedy algorithm that uses a geometric distribution with a nested logarithmic exponent. The level of nesting is optimized for each document model until no further improvement can be made. Thus, curve-fitting and Good-Turing estimate together provide us with the first smoothing step towards building a suitable language model for a document.

Table 1. A Typical Term Distribution for a Large Corpus

tf	N_{tf}	tf	N_{tf}
0	74,671,100,100	5	68,379
1	2,018,046	6	48,190
2	449,721	7	35,709
3	188,933	8	27,710
4	105,668	9	22,280

Using a smoothed function $S(N_{tf})$ for $E(N_{tf})$, the probability of a term t with frequency tf in document d can be computed as follows:

$$P_{GT}(t \mid d) = \frac{(tf+1)S(N_{tf+1})}{S(N_{tf})N_d}$$

3.2 Expanding a Document Model with the Corpus Model

A document model is not stable in the sense that there is a large number of missing terms, and there can be anomaly distributions of certain known terms. Using the same Good-Turing estimate, we can also build a model for the entire corpus. Such a corpus model will be stable, since it is obtained from a large number of documents. Furthermore, it can help us differentiate the contributions of different missing terms in a document. In a corpus about information retrieval, for example, the term “keyword” will likely happen more often than the term “crocodile”. As a result, it will be helpful to extend a document model with the corpus model.

There are two general ways of combining language models: the weighted sum approach (also called interpolation) and the weighted product approach:

$$P_{sum}(t \mid d) = \omega \times P_{document}(t \mid d) + (1-\omega) \times P_{corpus}(t)$$

$$P_{product}(t \mid d) = P_{document}(t \mid d)^{\omega} \times P_{corpus}(t)^{(1-\omega)}$$

where ω is a weighting parameter between 0 and 1. Both methods have been used in statistical language modeling for information retrieval. The advantage of the weighted sum is that the resulting probabilities are normalized, that is, the total probability mass for the combined model is still equal to one, whereas the weighted product is not normalized.

As will be discussed in section 5, there are two important differences between our model and the other language models. First, we acquire document models independently according to different sources and intentionally keep these models separate. Second, we consider generally ways of combining different models of a document by introducing parameters, which can be further individualized and optimized for each document. We believe that such a fine-tuning step for each document is an attractive feature of our model, since it has a potential in further improving the retrieval performance and is not available in most of the existing retrieval models.

3.3 Modeling a Query as a Sequence of Terms

With reasonably smoothed document models, we can now consider the process of query generation. This time we have a choice of treating a query as a set of terms or a sequence of terms. Treating a query as a set of terms is common in information retrieval and is also used in Ponte and Croft’s model [10]:

$$P_{set}(Q \mid d) = \prod_{t \in Q} P(t \mid d) \times \prod_{t \notin Q} (1.0 - P(t \mid d))$$

Here, the first part is the probability of producing the terms in a query, and the second part is the probability of not producing other terms. Including the second part in the evaluation is important, since the probabilities for each document model are not normalized.

Alternatively, we can treat a query as a sequence of terms. Each term is viewed as an independent event, and the query as the joined event. As a result, we can get the query probability by multiplying the individual term probabilities.

$$P_{sequence}(Q \mid d) = \prod_{i=1}^{m} P(t_i \mid d)$$

where t_1, t_2, …, t_m is the sequence of terms in query Q. This is our choice for two reasons. First, by treating a query as a sequence of terms, we will be able to handle the duplicate terms in a query. Of course, one can introduce weights into the set treatment of a query, but that will complicate the computation. Secondly, we want to be able to model phrases with local contexts, and this can only be done by viewing a query as a sequence of terms. This implies that we need to consider bigrams and possibly trigrams as well in our general language model for information retrieval.

3.4 Combining the Unigram Model with the Bigram Model

The combination of unigrams and bigrams is commonly handled through interpolation in statistical language modeling:

$$P(t_{i-1}, t_i \mid d) = \lambda_1 \times P_1(t_t \mid d) + \lambda_2 \times P_2(t_{i-1}, t_i \mid d)$$

where $\lambda_1 + \lambda_2 = 1$. This formula can be easily extended to include trigrams. The general idea is that by choosing appropriate values for λ’s, we will not lose information and the performance may be further improved. In fact, such interpolations can be modeled by hidden-Markov models, and there exists an automatic procedure, called EM (Expectation Maximization), that will learn the λ’s from a training corpus [7]. Thus, we can potentially individualize and optimize the combination process by choosing suitable λ’s for each document model. In the following experiments, λ's are set empirically, but in future we intend to use the EM algorithm to optimize these parameters.

4. EXPERIMENTAL RESULTS AND DISCUSSIONS

To demonstrate the effectiveness of our general language model, we conducted experiments on two test collections. The Wall Street Journal (WSJ) data is a medium-sized homogeneous collection, with over 250 megabytes of information and 74,520 documents. The TREC4 data is a large heterogeneous collection, with over 2 gigabytes of information and 567,529 documents. The TREC4 queries were used for evaluating both collections, since WSJ is actually part of the TREC4 collection. Table 2 below lists the detailed statistics of these two test collections.

Four different retrieval systems are used in our experiments. The baseline is the INQUERY system, and for the purpose of comparison, we also implemented Ponte and Croft's language model (LM). GLM(40) is the unigram version of our general language model, where the combination between a document model and the corpus model is handled through a weighted sum, with the weighting parameter set to be 40% for the document model. GLM2(40+90) is the combined model for unigrams and bigrams, with the weighting parameter between a document model and the corpus model set to be 40%, and the weighting parameter for the bigram model set to be 90%.

Table 2. Statistics of the WSJ and TREC4 Data Sets

Collections	WSJ	TREC4
Data Size	253+ MB	2+ GB
#Documents	74,520	567,529
#Queries	49	49
Term Types	119,854	466,651
Term Tokens	20,061,761	144,714,632
Pair Types	5,606,265	23,654,736
Pair Tokens	19,987,241	144,147,212

As shown in table 3, the results of all the language models are comparable to that of INQUERY. In addition, our unigram model GLM(40) did 8.44% better than Ponte and Croft's LM and our combined unigram and bigram model GLM(40+90) did 16.38% better than LM. This is a clear indication that phrases of word pairs can be useful in improving the retrieval performance in the context of language modeling.

Table 3. Experimental Results on the WSJ Data Set

Retrieval Methods	11-pt Average	%Change	%Change
INQUERY	0.2172	-	
LM	0.2027	- 6.68%	-
GLM(40)	0.2198	+ 1.20%	+ 8.44%
GLM2(40+90)	0.2359	+ 8.61%	+ 16.38%

For the large TREC4 data set, the results of the language models are once again comparable to that of INQUERY, as shown in table 4. However, the improvement of our models over Ponte and Croft's LM is not as significant as that for the WSJ data set. This is probably due to the heterogeneous nature of the TREC4 collection. In Ponte and Croft's model, there is a pathological problem in using the default probability: missing terms with relatively high frequencies in a corpus are often assigned with high default values, which could be problematic for a homogeneous collection, but less serious for a heterogeneous collection. Nevertheless, our models still did better than Ponte and Croft's and the word pairs are still shown to be useful in improving the retrieval performance.

Note that in [10], a significant improvement of LM model over INQUERY was reported for the TREC4 data set. This is, however, not observed in our experiments. One reason for this difference may be the variation in preprocessing the raw TREC4 data set (e.g., different choices of SGML sections, stop words, and stemming). A similar comment was also made in [8] regarding the differences in retrieval performance.

Table 4. Experimental Results on the TREC4 Data Set

Retrieval Methods	11-pt Average	%Change	%Change
INQUERY	0.1917	-	
LM	0.1890	- 1.41%	-
GLM(40)	0.1905	- 0.63%	+ 0.79%
GLM2(40+90)	0.1923	+ 0.31%	+ 1.75%

The 11-point average precision represents the average case. To see the detailed changes, we also plot the precision values at the different recall levels. As shown in figure 1, our language models improve the performance consistently across all the recall levels. Note that our model has much potential for further improvement, since all the combination parameters can be individualized and optimized instead of setting them to be the same for all the documents.

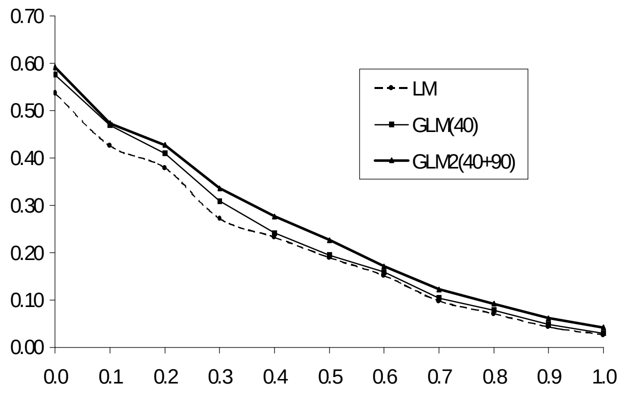

Figure 1. Detailed Results on the WSJ Data Set

5. COMPARISONS TO OTHER LANGUAGE MODELS

Our model is similar to Hiemstra's model [5] in that we also use interpolation to expand a document model with the corpus model. The difference is that Hiemstra did not smooth each document model before the interpolation. In addition, Hiemstra used the inverted document frequency (idf) to estimate a term probability in the corpus. This allows him to relate his model to the well-known tf-idf formulation. However, the idf-based estimation will loose information about the corpus, since a term may appear different times in different documents.

Our work was originally motivated by Ponte and Croft's model [9, 10]. However, instead of combining a document model with the corpus model in an ad-hoc manner, we intentionally keep them separate and consider general ways of combining them. In Ponte and Croft's model, the corpus term probabilities are used as default for those terms that do not appear in a document. One potential problem is that some missing terms (i.e., a word with the characteristics of a stopword) may be assigned with higher scores than the terms that actually appear in the document. Such a pathological problem was recognized by Ponte and Croft, and was left as future work. In cases when the terms do appear in a document, they introduced the average probabilities to further smooth a document model. The probability of a term in a specific document is combined with the average probability of the term through a geometric distribution (called the risk factor). The advantage of this approach is that the risk factor is individualized for each term, but the disadvantage is that it is ad hoc and leaves little room for optimization. Finally, a query is treated as a set of terms rather than a sequence of terms, mostly due to the fact that the probabilities are not normalized. Although the frequency information can be added by introducing weights to different terms, that will complicate the process and make it difficult to expand for phrases.

The BBN model [6, 8] uses a simple hidden-Markov model for combining a document model and the corpus model. No smoothing was used for individual documents, but the weighting parameter between a document model and the corpus model (called the general environment) can be trained and optimized through a learning procedure. For the TREC data, they can use the relevance judgement in the previous years for training, but such training data may not be easily available for other data sets. Nevertheless, an idea similar to this may be borrowed in our language model to individualize and optimize the weighting parameters between a document model and the corpus model.

6. CONCLUSIONS AND FUTURE WORK

We have proposed a simple yet intuitive language model for information retrieval. It is based on a range of data smoothing techniques, including the Good-Turing estimate, curve-fitting functions, and model combinations. Models obtained from different sources are intentionally kept separate, and as a result, our model is easy to understand and expand. We also conducted experiments on two test collections, and the results showed that the performance of our model is comparable to that of INQUERY and better than that of Ponte and Croft's language model. In particular, word pairs are shown to be useful in improving the retrieval performance. Our model can potentially be improved by individualizing and optimizing the parameters for combining models of different sources. Furthermore, our model is rooted on the solid foundation of statistical natural language processing. Any new techniques developed for data smoothing can be easily incorporated into our model. In this sense, our model serves as a general framework for language-based information retrieval.

For the future work, we are planning to add a number of extensions to our language model for information retrieval. First of all, we can individualize and optimize the combination of the unigram and bigram models of a document through the hidden-Markov training algorithm. Secondly, we can identify word pairs within different boundaries of a document such as paragraphs or sentences. In the current experiments, we simply take all the word pairs in a document, which may not be meaningful for those pairs that cross sentence boundaries. Thirdly, we can explore automatic methods for combining a document model with the corpus model so that the weighting parameters can be individualized and optimized. Finally, due to the simplicity of our model, it opens up the door for many other extensions, such as relevance feedback (also discussed in [9]), syntactic preprocessing, and possibly sense-based information retrieval. In particular, the syntactic preprocessing may involve spelling correction, phrase normalization (e.g., convert "information retrieval of biological science" and "retrieval of biological science information" to "biological science information retrieval"), and possibly syntactic parsing to selectively apply higher order N-grams to some longer phrases.

ACKNOWLEDGMENTS

The authors would like to thank Jay Ponte, Stephen Harding, Margaret Connell, Mark Sanderson, Jinxi Xu, Jeremy Pickens, and Lisa Ballesteros for their help and support.

This material is based on work supported in part by the National Science Foundation, Library of Congress and Department of Commerce under cooperative agreement number EEC-9209623. Any opinions, findings and conclusions or recommendations expressed in this paper are the authors and do not necessarily reflect those of the sponsors.

REFERENCES

[1] Callan, J.P., Croft, W.B., and Broglio, J. TREC and TIPSTER experiments with INQUERY. Information Processing and Management, 31(3): 327-343, 1995.

[2] Charniak, E. Statistical Language Learning. The MIT Press, Cambridge MA, 1993.

[3] Croft, W.B., and Turtle, H.R. Text Retrieval and Inference. In Text-Based Intelligent Systems, edited by Paul S. Jacob, pages 127-155, Lawrence Erlbaum Associates, Publishers, 1992.

[4] Frakes, W.B., and Baeza-Yates, R. (editors). Information Retrieval: Data Structures and Algorithms. Englewood Cliffs, New Jersey: Prentice Hall, 1992.

[5] Hiemstra, D. A Linguistically Motivated Probabilistic Model of Information Retrieval. Second European Conference on Digital Libraries, pages 569-584, 1998.

[6] Leek, T., Miller, D.R.H., and Schwartz, R.M. A Hidden Markov Model Information Retrieval System. TREC-7 Proceedings, 1998.

[7] Manning, C., and Schütze, H. Foundations of Statistical Natural Language Processing. The MIT Press, 1999.

[8] Miller, D.R.H., Leek, T., and Schwartz, R.M. A Hidden Markov Model Information Retrieval System. In Proceedings of SIGIR'99, pages 214-221. University of California, Berkeley, Aug., 1999.

[9] Ponte, J.M. A Language Modeling Approach to Information Retrieval. Ph.D. thesis, University of Massachusetts at Amherst, 1998.

[10] Ponte, J.M., and Croft, W.B. A Language Modeling Approach to Information Retrieval. In Proceedings of SIGIR'98, pages 275-281. Melbourne, Australia, 1998.

[11] Robertson, S.E. The probability ranking principle in IR. Journal of Documentation, 33(4): 294-304, December 1977.

[12] Salton, G. Automatic Information Organization and Retrieval. McGraw-Hill, 1968.