An Efficient Computation of the Multiple-Bernoulli Language Model

Leif Azzopardi[1] and David E. Losada[2] *

[1] Information Language Processing Group, University of Amsterdam, Netherlands
leif@science.uva.nl
[2] Grupo de Sistemas Inteligentes, Departamento de Electrónica y Computación,
Universidad de Santiago de Compostela, Spain
dlosada@dec.usc.es

Abstract. The Multiple Bernoulli (MB) Language Model has been generally considered too computationally expensive for practical purposes and superseded by the more efficient multinomial approach. While, the model has many attractive properties, little is actually known about the retrieval effectiveness of the MB model due to its high cost of execution. In this paper, we show how an efficient implementation of this model can be achieved. The resulting method is comparable in terms of efficiency to other standard term matching algorithms (such as the vector space model, BM25 and the multinomial Language Model).

1 Introduction

The Multiple-Bernoulli Language Model was originally proposed for Information Retrieval (IR) in [3] and has been recently extended in the context of Bayesian Learning[2, 1]. The MB language model is an appealing IR model, providing a coherent framework in which queries and documents are treated in a uniform manner. Also, the model provides implicit length normalization, because documents which contain many non-query terms are penalized for being off topic. This feature would suggest that the applicability of the model would favor particular IR tasks where length normalization is critical (such as in element retrieval). Already, some evidence to this tune has been shown in the context of sentence retrieval[1].

Although the computational complexity of the MB model could be thought to be too high for the model to be implemented in a practical setting, we show here that an efficient method can be designed to do retrieval efficiently. If we examine the formulation of the MB model in Eq. 1, we can see that the probability of generating a query q given the document model θ_d involves a computation across

* The second author is supported by the "Ramón y Cajal" R&D program, which is funded in part by "Ministerio de Educación y Ciencia" and in part by FEDER funds. This work was financially supported by "Ministerio de Educación y Ciencia" through research project ref. TIN2005-08521-C02-01.

all terms t_i in the vocabulary T, (i.e. $t_i \in T$).

$$p(q|\theta_d) = \prod_{t_i \in q} p(t_i|\theta_d) \prod_{t_i \notin q} (1 - p(t_i|\theta_d)) \tag{1}$$

In the worse case scenario a direct implementation for a collection of documents would be equal to the number of terms in the vocabulary, $|T|$ multiplied by the number of documents in the collection, $|D|$ (i.e. $|T| \times |D|$).

2 Optimization

Instead of directly computing $p(q|\theta_d)$ for every d in the collection of document D, an optimization of the model is possible, by decomposing the scoring procedure. First, we require a pre-computation given the set of model parameters which define θ_d, before query time. This estimates the probability of an empty query being generated from the document model. Then, at query time, the pre-computed document score is adjusted according to the terms that appear in the query. To facilitate the optimization, we shall require some extra definitions. Let q_e be the empty query, and let d_e be an empty document, where the number of times t_i occurs in the document is zero for any t_i (denoted as $n(t_i, d_e) = 0$). The document model of d_e is θ_{d_e} and its use shall become apparent shortly.

2.1 Pre-Computation before Query Time:

The probability of the empty query q_e given each document model θ_d is computed offline.

$$p(q_e|\theta_d) = \prod_{t_i \notin q_e} (1 - p(t_i|\theta_d)) \tag{2}$$

Since the query is empty, this involves a product across all vocabulary terms. Whilst this value is document dependent, we can design an efficient method for computing the $p(q_e|\theta_d)$. This is accomplished by first scoring a hypothetical empty document and then updating this score given the terms seen in the actual document. Thus, we pre-compute the probability of the empty query given the empty document model, $p(q_e|\theta_{d_e})$ as follows:

$$p(q_e|\theta_{d_e}) = \prod_{t_i \notin q_e} (1 - p(t_i|\theta_{d_e})) \tag{3}$$

Note that any term t_i is unseen in the empty document and, therefore, the value $p(t_i|\theta_{d_e})$ is computed using $n(t_i, d_e) = 0$. Once we see the actual document d we can compute the probability of producing the empty query, $p(q_e|\theta_d)$, starting from $p(q_e|\theta_{d_e})$. The approach can be illustrated as follows. Starting from $p(q_e|\theta_{d_e})$ can be thought as an initial assumption that any document is empty. As we see the actual document terms, we update the probability score, removing $(1 - p(t_i|\theta_{d_e}))$ which was computed assuming $n(t_i, d) = 0$. And then multiplying by

$(1 - p(t_i|\theta_d))$, which is computed using the actual term document counts (i.e. $n(t_i, d) > 0$). Formally,

$$p(q_e|\theta_d) = p(q_e|\theta_{d_e}) \cdot \prod_{t_i \in d} \frac{1 - p(t_i|\theta_d)}{1 - p(t_i|\theta_{d_e})} \tag{4}$$

That is, we only need to go on the seen terms whereas the unseen terms take its probability from the pre-computed $p(q_e|\theta_{d_e})$, which needs only to be computed once. This is the first significant saving because the number of unique terms in the documents is usually several orders of magnitude less than the size of the vocabulary, $|T|$. The reader may note that this imposes an implicit constraint on the optimization, because of the assumption that $p(t_i|\theta_d) = p(t_i|\theta_{d_e})$ for all the terms which are unseen in the document d. This equality holds in the original Ponte and Croft formulation [3] as the unseen terms' probabilities are assumed to be equal to the probability in a background model. That is, there is no document dependent factor in the unseen term probability. On the other hand, in the context of Bayesian Learning, the case is slightly different. The basic MB formulation of the $p(t_i|\theta_d)$ formula in [2] and [1] also depend only on background probabilities for unseen terms and, therefore, the same efficient approach can be taken. However, in a variation of the MB model to deal with non-binary term-document counts (called Model B in [2]), the above method cannot be immediately applied as $p(t_i|\theta_d)$ is not equal to $p(t_i|\theta_{d_e})$ because the final term estimate is proportional to the length of a document. When this is the case, then a generalization of the process designed here can be employed. Instead of assuming one hypothetical document, which is empty (i.e. $\sum_t n(t, d_e) = 0$), a set of hypothetical documents need to be constructed, where the length of each hypothetical document is $1, \ldots, n$, n being the document length of the largest document[3]. This enables the computation to be performed almost as efficiently, but incurs higher storage/memory costs.

2.2 Computation at Query Time

For each query term we adjust the contribution from the query terms in the empty document model (eq. 5). Next, we compute the factor involving the query-document matching terms (eq. 6).

$$p(q|\theta_d) = p(q_e|\theta_d) \cdot \prod_{t_i \in q} \frac{p(t_i|\theta_{d_e})}{1 - p(t_i|\theta_{d_e})} \tag{5}$$

$$\times \prod_{t_i \in q \cap d} \frac{p(t_i|\theta_d)}{p(t_i|\theta_{d_e})} \cdot \frac{1 - p(t_i|\theta_{d_e})}{1 - p(t_i|\theta_d)} \tag{6}$$

Note that the product across query terms in eq. 5 is document independent and, thus, it only needs to be computed once for each query. The product across matching terms in eq. 6 introduces the right score for a matching term, $p(t_i|\theta_d)$,

[3] Actually, only one hypothetical document is needed for each unique document length.

and removes the score introduced in the previous steps[4]. This speed up uses a similar tactic to that suggested in [4] for the multinomial approach. However, in the multinomial model unseen query terms are not considered and, hence, there is no need for an initial query score.

3 Complexity Analysis

We described the computation complexity according to the number of term score calculations are required. The before query time pre-computation of calculating $p(q_e|\theta_{d_e})$ (eq. 3) takes $|T|$ steps and to compute the value $p(q_e|\theta_d)$ for all the documents (eq. 4) takes $|D| \cdot |T| \cdot s$ steps, where s is the sparsity expressed as the percentage of non-zero entries in the document-term matrix. This is computed offline and so does not directly affect on-line performance. At query time, the online computations in eq. 5 involves $|q|$ steps, and eq. 6 takes $|d| \cdot |q| \cdot s$ iterations, where $|q|$ is the number of query terms. Under this optimization a very significant reduction in the run time of the MB retrieval model can be achieved which makes it comparable to other state of the art retrieval models.

4 Conclusion

We have presented an efficient method for computing the MB model, which reduces significantly the expected matching time[5]. From prior research and our own intuitions we believe that the MB model will be more effective in specific retrieval scenarios, such as when the elements to be retrieved are short and need to be focused or when the variation in size of retrievable elements is high. Further work will be directed at identifying retrieval scenarios that can exploit the attractive properties of the MB model.

References

1. D. E. Losada. Language modeling for sentence retrieval: a comparison between multiple-bernoulli models and multinomial models. In *Information Retrieval and Theory Workshop*, Glasgow, UK, 2005.
2. D. Metlzer, V. Lavrenko, and W. B. Croft. Formal multiple-bernoulli models for language modeling. In *Proc. 27th ACM Conference on Research and Development in Information Retrieval, SIGIR'04*, pages 540–541, Sheffield, UK, 2004. ACM press.
3. J. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proc. 21st ACM Conference on Research and Development in Information Retrieval, SIGIR'98*, pages 275–281, Melbourne, Australia, 1998.
4. C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, 22(2):179–214, 2004.

[4] Note that, for a matching term, after applying eqs 3, 4 and 5, we have a contribution equal to $\frac{p(t_i|\theta_{d_e})\cdot(1-p(t_i|\theta_d))}{(1-p(t_i|\theta_{d_e}))}$. We just multiply by the inverse of this value.

[5] We have implemented the MB proposed in [1] in LEMUR 4.0, and this code will be made freely available.