A Machine Learning Approach for Improved BM25 Retrieval

Krysta M. Svore
Microsoft Research
One Microsoft Way
Redmond, WA 98052
ksvore@microsoft.com

Christopher J. C. Burges
Microsoft Research
One Microsoft Way
Redmond, WA 98052
cburges@microsoft.com

ABSTRACT

Despite the widespread use of BM25, there have been few studies examining its effectiveness on a document description over single and multiple field combinations. We determine the effectiveness of BM25 on various document fields. We find that BM25 models relevance on popularity fields such as anchor text and query click information no better than a linear function of the field attributes. We also find query click information to be the single most important field for retrieval. In response, we develop a machine learning approach to BM25-style retrieval that learns, using LambdaRank, from the input attributes of BM25. Our model significantly improves retrieval effectiveness over BM25 and BM25F. Our data-driven approach is fast, effective, avoids the problem of parameter tuning, and can directly optimize for several common information retrieval measures. We demonstrate the advantages of our model on a very large real-world Web data collection.

Categories and Subject Descriptors: H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval I.2.6 [Artifical Intelligence]: Learning

General Terms: Algorithms, Experimentation, Theory.

Keywords: Web Search, Retrieval Models, Learning to Rank, BM25.

1. INTRODUCTION

BM25 [15] is arguably one of the most important and widely used information retrieval functions. BM25F [16] is an extension of BM25 that prescribes how to compute BM25 across a document description over several fields. A challenge to using BM25 and BM25F is the necessity of tuning $2K+1$ parameters for a document description over K fields. Tuning can be accomplished using grid-search or gradient descent [21]. Each method has its drawbacks; grid-search can be prohibitively slow when the data collection is large, while gradient descent [21] is much faster but does not optimize parameters directly for a target measure.

CIKM'09, November 2–6, 2009, Hong Kong, China.

Recently, it has been shown that LambdaRank [2] is empirically optimal [7, 25] for NDCG and other IR measures. We could extend the approach in [21] to use LambdaRank to optimize the BM25 parameters for a chosen IR measure, but the function is still restricted to the BM25 model. Instead, we build upon LambdaRank to develop a machine learning approach to BM25-style retrieval. Our model can be used as a framework for learning other functions and offers value in the design of future information retreival systems. Our primary contributions are threefold (see [20] for details): (1) We empirically determine the effectiveness of BM25 for different fields. Although BM25 is effective on the title and URL fields, we find that on popularity fields it does not perform as well as a linear model. (2) We develop a data-driven machine learning model called LambdaBM25 that is based on the attributes of BM25 [15] and the training method of LambdaRank [2]. Our model is both fast and simple; it does not require parameter tuning and is an extension of a state-of-the-art ranking approach. It also directly optimizes for several IR measures [7, 25]. (3) We extend our empirical analysis to a document description over various field combinations. We confirm that BM25F [16] is better than a linear function of BM25 scores. We extend our model to document descriptions over field combinations and find it consistently outperforms BM25F with statistical significance.

2. RELATED WORK

There have been a number of approaches to document retrieval ranging from simple to complex models [12, 13, 22]. BM25 [15] is based on a probabilistic information retrieval model [19] which incorporates attributes of documents, such as term frequencies, document frequencies, and document length. BM25F is a simple extension of BM25 for combining attributes across multiple fields [16]. A drawback of BM25 and BM25F is the difficulty in optimizing the function parameters for a given information retrieval measure. There have been extensive studies on how to set term frequency saturation and length normalization parameters [17, 9, 21].

Recent studies demonstrate the effectiveness of query click data for ranking [1, 6, 8, 24]. However, to our knowledge, there is no detailed study of the effectiveness of BM25 on single document fields or on subsets of document fields, including anchor text and query click logs. In addition, we are unaware of efforts to develop a directly analagous retrieval model based on the same attributes as BM25. Our work provides both an extensive study of the contributions of different document fields to information retrieval and a framework for improving BM25-style retrieval.

3. DOCUMENT FIELDS

A Web *document description* is composed of several *fields* of information. Field information is preprocessed by removing punctuation, converting to lowercase, and removing html markup. We consider a query q to be composed of at most 10 terms. The document frequency for term t is the number of documents in the collection that contain term t in their document descriptions. Term frequency is calculated per term and per field by counting the number of occurrences of term t in field F of the document. Field length is the number of terms in the field.

Content fields include the body text (the html content of the page), the document's title (indicated through html <TITLE> tags), and the word-broken URL text. *Popularity fields* include anchor text and query click information. Unlike content fields, popularity fields are not written or controlled by the document's owner, but rather are an aggregation over information about the page from many authors. The anchor text field is composed of the text of all incoming links to the page. We denote the field as a set of unique anchor text strings and the corresponding numbers of incoming links with that string. The query click field is built from query session data [1, 8, 11] (see [8] for details) extracted from one year of a commercial search engine's query log files and is represented by a set of query-score pairs $(q, Score(d,q))$, where q is a unique query string and $Score(d,q)$ is derived from raw session click data as

$$Score(d,q) = \frac{C(d,q,click) + \beta * C(d,q,last\_click)}{C(d,q)}, \quad (1)$$

where $C(d,q)$ is the number of times d is shown to the user when q is issued, $C(d,q,click)$ is the number of times d is clicked for q, and $C(d,q,last\_click)$ is the number of times d is the temporally last click of q. β is a scaling factor and can be tuned. The term frequency of term t for the query click field is calculated as $\sum_{p|t\in p} Score(d,q)$, where p is the set of query-score pairs.

4. BM25

BM25 [15, 19] is a function of term frequencies, document frequencies, and the field length for a single field. BM25F [16] is an extension of BM25 to a document description over multiple fields and reduces to BM25 when calculated over a single field; we refer to both functions as BM25_F, where F is a specification of the fields contained in the document description.

BM25_F is computed for document d with description over fields F and query q as: $S = \sum_{t\in q} TF_t * I_t$. The sum is over all terms t in query q. I_t is the Robertson-Sparck-Jones inverse document frequency of term t:

$$I_t = \log \frac{N - df + 0.5}{df + 0.5}, \quad (2)$$

where N is the number of documents in the collection, df is the document frequency of term t. We calculate document frequency over the body field for all document frequency attributes[1]. TF_t is a term frequency saturation formula: $TF_t = \frac{f}{k+f}$, where f is calculated as

$$f = \sum_F \frac{w_F * tf_F}{\beta_F}. \quad (3)$$

tf_F is the term frequency attribute of term t in field F, k is the saturation parameter, and w_F is a field weight parameter. β_F accounts for varying field lengths: $\beta_F = (1 - b_F) + b_F(\ell_F / avg\ell_F)$, where b_F is a parameter between 0 and 1, ℓ_F is the length of the field, and $avg\ell_F$ is the average length of the field in the document collection.

BM25_F requires the tuning of $2K + 1$ parameters, when calculated across K fields, namely k, b_F, and w_F. Tuning can be done using grid-search or gradient descent [21]. In our experiments, we tuned the parameters of BM25_F using grid search over 10K queries and for $K > 3$, it took over 2 weeks to complete on an Intel Xeon 2.93GHz processor with 127GB of RAM.

5. LEARNING A BM25-STYLE FUNCTION

We now describe our simple machine learning ranking model that uses the input attributes of BM25_F and the training method of LambdaRank. LambdaRank [2] is a state-of-the-art ranking algorithm that optimizes for IR measures. For complete details, see [2]. LambdaRank is both a list-based and a pair-based neural network learning algorithm and is an extension of RankNet [3]; it is trained on pairs of documents per query, where documents in a pair have different relevance labels. In most machine learning tasks, a target evaluation measure is used for evaluation and an optimization measure, generally a smooth approximation to the target measure, is used to train the system. Typical IR target costs are either flat or non-differentiable everywhere, thus direct optimization of the target measure is quite challenging. LambdaRank [2] leverages the fact that neural net training only needs the gradients of the measure with respect to the model scores, and not the function itself, thus avoiding the problem of direct optimization. The gradients are defined by specifying rules about how swapping two documents, after sorting them by score for a given query, changes the measure. The gradient definition is general and can work with any target evaluation measure.

There are several challenges to using BM25 despite its strong retrieval capacity, including the requirement of parameter tuning, the inability to directly optimize for an IR measure, and the restrictions of the underlying probabilistic model. We directly address these challenges by introducing a new machine learning approach to BM25-like retrieval called LambdaBM25, which is trained over a large data collection using LambdaRank due to its flexibility, ease of training, and state-of-the-art ranking accuracy. BM25 can be prohibitively expensive when trained on a document description over many fields, especially with the growing use of anchor text, click information, and other metadata. LambdaBM25 does not require parameter tuning since the function is learned directly from the train collection and can optimize for several IR measures [7, 25].

LambdaBM25 has the flexibility to learn complex relationships between attributes directly from the data, for example if documents tend to be verbose or elaborative, while BM25 is limited to a predefined probabilistic model. Our model has the additional advantage that it does not require that the attributes be statistically independent, as in [19].

[1]We also used the whole document description, but found little difference in accuracy over using only the body field.

We recognize that in learning our model directly from a large data collection, we lose the probabilistic interpretation inherent to BM25. However, our model has an additional advantage in that it is very flexible, and can be extended to include other fields in the document description as new fields become available.

We develop our model as follows. We train our model using LambdaRank and the same input attributes as BM25, namely term frequency, document frequency, and field length, for each field included in the document description. Although we could include additional attributes, we would like to maintain a fair comparison to the BM25 retrieval function because it is so widely used. We train single- and two-layer LambdaRank neural nets to optimize for NDCG with varying numbers of hidden nodes chosen using the validation set.

6. DATA AND EVALUATION

We evaluate our method on a real-world Web-scale data collection containing English queries sampled from query log files of a commercial search engine and corresponding URLs. We perform stopword removal and some stemming on queries. Our train/validation/test data contains 67683 /11911/12185 queries, respectively. Each query is associated with on average 150-200 documents (URLs) together with a vector of feature attributes extracted for the query-URL pair. Each query-URL pair has a human-generated label between 0, meaning d is not relevant to q, and 4, meaning document d is the most relevant to query q and 0.

We evaluate using Normalized Discounted Cumulative Gain (NDCG) [10] at truncation levels 1, 3, and 10. Mean NDCG is defined for query q as

$$\text{Mean NDCG@}L = \frac{100}{N * Z} \sum_{q=1}^{N} \sum_{r=1}^{L} \frac{2^{l(r)} - 1}{\log(1 + r)} \qquad (4)$$

where N is the number of queries, $l(r) \in \{0, \ldots, 4\}$ is the relevance label of the document at rank position r and L is the truncation level to which NDCG is computed. Z is chosen such that the perfect ranking would result in NDCG@L_q = 100. A significant difference should be read as significant at the 95% level using a t-test.

7. EXPERIMENTS AND RESULTS

In all experiments, the parameters of $BM25_F$ are tuned to optimize NDCG@1 on our validation set (it was prohibitively slow to tune on the train set) using a $2K$-D grid search as in [21]; we consider 1000 epochs or convergence of NDCG@1 as the stopping criterion. Parameters found are listed in the extended technical report [20]. We also tried an approach similar to the gradient-based approach in [21] and found results to be almost identical.

We first determine which single field is the most effective in terms of ranking using $BM25_F$. In the upper first three columns of Table 1, we report results for $BM25_F$ on a document description restricted to a single field. We find Title (T), URL (U), and Body (B) are equally effective and popularity fields achieve higher NDCG. In particular, the query click field achieves the highest NDCG accuracy.

We next compare $BM25_F$ to single-layer $LambdaBM25_F$ on a single field F. Since $BM25_F$ is highly nonlinear, we expect it to outperform a simple linear combination of input attributes. Our linear model cannot, for example, divide term frequency by document frequency or field length; these two operations have been shown to give improved retrieval accuracy [19]. In all experiments, we choose the best training epoch and learning rate based on the validation data which is a learning rate of 10^{-5} and 500 epochs.

Table 1 contains results for single-layer $LambdaBM25_F$. For content fields, we find that $BM25_F$ is significantly better than a linear combination of input attributes since $BM25_F$ was designed for improved accuracy over a linear term frequency function when using content fields. For popularity fields, on the other hand, our single-layer $LambdaBM25_F$ model performs similar to or better than $BM25_F$. Such results were hypothesized in [21]; since popularity fields draw content from authors other than the document's owner, it seems reasonable that the BM25 function, which was built for content fields, may not model the data much better than a linear function of input attributes.

Finally, we train our two-layer $LambdaBM25_F$ model and determine if it can outperform $BM25_F$. Results are shown in Table 1. We find the following numbers of hidden nodes to be best: Title (10), URL (15), Body (15), Anchor (15), Click (5). We find that for the Body, Anchor, and Click fields, $LambdaBM25_F$ outperforms significantly $BM25_F$; $BM25_F$ appears to model short, succinct, non-repeatable fields well, but fails to model longer fields with similar accuracy. As the length of the field grows, it is beneficial to learn richer relationships between term frequency, document frequency, and field length, which $LambdaBM25_F$ is able to do.

We next seek to determine the most effective combination of fields to include in the document description for $BM25_F$. The first three lower columns of Table 1 list the results of $BM25_F$ on various field combinations. We find that using multiple fields in the document description is superior to using a single field, unless that single field is the query click field; the only combination of fields to outperform $BM25_C$ are combinations that include the query click field. Note that the addition of anchor text to the C,U,T,B combination yields an insignificant improvement in accuracy, but when query click information is not available, the anchor field yields significant accuracy improvement between the U,T,B and A,U,T,B field combinations.

To determine if $BM25_F$ is better that a linear function of input attributes, we learn single-layer $LambdaBM25_F$ models for each combination of fields. As shown in the lower middle columns of Table 1, we find that $BM25_F$ performs as well or better than single-layer $LambdaBM25_F$ for all field combinations; our results confirm that a linear combination of fields is insufficient for good retrieval accuracy [16].

Finally, we train our two-layer $LambdaBM25_F$ models using 15 hidden nodes. For every field combination, as shown in Table 1, $LambdaBM25_F$ achieves gains with statistical significance over the corresponding $BM25_F$ model. For combinations that include popularity fields, we see even more substantial gains over $BM25_F$.

8. CONCLUSIONS AND FUTURE WORK

Our main contribution is a new information retrieval model trained using LambdaRank and the input attributes of BM25 called $LambdaBM25_F$, which significantly improves retrieval effectiveness over $BM25_F$ for most single-field, in particular popularity fields, and ***all multiple-field document descriptions***. $LambdaBM25_F$ optimizes directly for the chosen target IR evaluation measure and avoids the necessity

Table 1: Mean NDCG accuracy results on the test set for $BM25_F$, 1-layer $LambdaBM25_F$, and 2-layer $LambdaBM25_F$ for single fields and multiple field combinations. Statistical significance is determined at the 95% confidence level using a t-test. Bold indicates statistical significance over the corresponding $BM25_F$ model. Italic indicates statistical significance of the corresponding $BM25_F$ model over the $LambdaBM25_F$ model. Parentheses indicate no statistically significant difference.

Field(s)	$BM25_F$			$LambdaBM25_F$, 1-Layer			$LambdaBM25_F$, 2-Layer		
	@1	@3	@10	@1	@3	@10	@1	@3	@10
T	24.50	27.23	33.32	*20.79*	*24.93*	*32.51*	(24.31)	(27.38)	**33.86**
U	24.96	27.24	32.77	*22.96*	*26.38*	33.17	*23.69*	*26.70*	**33.21**
B	24.35	27.92	35.07	*18.03*	*21.93*	*30.60*	**27.53**	**30.49**	**37.03**
A	33.50	32.53	33.37	(33.83)	**33.11**	**34.73**	**36.33**	**34.68**	**35.33**
C	40.07	36.62	35.89	*39.34*	(36.50)	(35.96)	**41.61**	**38.01**	**37.19**
T, B	27.84	30.81	36.98	*25.42*	*28.81*	*35.80*	**29.61**	**32.49**	**38.93**
U, T, B	30.81	33.30	39.53	*29.28*	*32.08*	*38.75*	**34.26**	**37.03**	**43.05**
A, U, T, B	38.66	38.83	43.42	(38.91)	(38.84)	*42.81*	**43.70**	**42.58**	**46.21**
C, U, T, B	45.29	43.37	46.83	*43.34*	*41.70*	*45.04*	**49.70**	**46.58**	**49.14**
C, A, U, T, B	45.41	43.53	46.88	*44.60*	*42.33*	*45.44*	**50.33**	**47.14**	**49.47**

of parameter tuning, yielding a significantly faster approach. Our model is general and can potentially act as a framework for modelling other retrieval functions.

In the future we would like to perform more extensive studies to determine the relative importance of attributes in our model. We would also like to determine the effectiveness of LambdaBM25 as a scoring function, where the scores can be used as inputs to a more complex ranking system, for example as a single feature in recent TREC retrieval systems [5, 4]. Finally, we plan to expand our model to learn proximity relationships and determine if incorporating such features can learn a better function than, for example, the proximity models given in [14, 18].

9. REFERENCES

[1] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 19–26, 2006.

[2] C. Burges, R. Ragno, and Q. Le. Learning to rank with nonsmooth cost functions. In *Advances in Neural Information Processing Systems (NIPS)*, 2006. See also MSR Technical Report MSR-TR-2006-60.

[3] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *International Conference on Machine Learning (ICML)*, Bonn, Germany, 2005.

[4] N. Craswell and D. Hawking. Overview of the TREC 2004 web track. In *Proceedings of TREC 2004*, 2004.

[5] N. Craswell, D. Hawking, R. Wilkinson, and M. Wu. Overview of the TREC 2003 web track. In *Proceedings of TREC 2003*, 2003.

[6] N. Craswell and M. Szummer. Random walk on the click graph. In *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2007.

[7] P. Donmez, K. Svore, and C. Burges. On the local optimality of LambdaRank. In *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2009.

[8] J. Gao, W. Yuan, X. Li, K. Deng, and J.-Y. Nie. Smoothing clickthrough data for web search ranking. In *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2009.

[9] B. He and I. Ounis. On setting the hyper-parameters of term frequency normalization for information retrieval. *ACM Transactions on Information Systems (TOIS)*, 25(3):13, 2007.

[10] K. Jarvelin and J. Kekalainen. IR evaluation methods for retrieving highly relevant documents. In *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 41–48, 2000.

[11] T. Joachims. Optimizing search engines using clickthrough data. In *SIGKDD*, pages 133–142, 2002.

[12] D. Metzler. Generalized inverse document frequency. In *ACM Conference on Information Knowledge Management (CIKM)*, 2008.

[13] P. Ogilvie and J. Callan. Combining document representations for known item search. In *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2003.

[14] Y. Rasolofo and J. Savoy. Term proximity scoring for keyword-based retrieval systems. In *Proceedings of the 25th European Conference on IR Research (ECIR)*, 2003.

[15] S. Robertson and S. Walker. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 345–354, 1994.

[16] S. Robertson, H. Zaragoza, and M. Taylor. Simple BM25 extension to multiple weighted fields. In *ACM Conference on Information Knowledge Management (CIKM)*, pages 42–49, 2004.

[17] A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. In *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 21–29, 1996.

[18] R. Song, M. Taylor, J.-R. Wen, H.-W. Hon, and Y. Yu. Viewing term proximity from a different perspective. *Advances in Information Retrieval, Lecture Notes in Computer Science*, 4956/2008:346–357, 2008.

[19] K. Sparck-Jones, S. Walker, and S. Robertson. A probabilistic model of information retrieval: development and comparative experiments. *Information Processing and Management*, 36:809–840, 2000.

[20] K. Svore and C. Burges. A machine learning approach for improved bm25 retrieval. *Microsoft Technical Report MSR-TR-2009-92*, 2009.

[21] M. Taylor, H. Zaragoza, N. Craswell, S. Robertson, and C. Burges. Optimisation methods for ranking functions with multiple parameters. In *ACM Conference on Information Knowledge Management (CIKM)*, 2006.

[22] R. Wilkinson. Effective retrieval of structured documents. In *Research and Development in Information Retrieval*, pages 311–317, 1994.

[23] Q. Wu, C. Burges, K. Svore, and J. Gao. Ranking, boosting and model adaptation. *Microsoft Technical Report MSR-TR-2008-109*, 2008.

[24] G. Xue, H.-J. Zeng, Z. Chen, Y. Yu, W.-Y. Ma, W. Xi, and W. Fan. Optimizing web search using web click-through information. In *ACM Conference on Information Knowledge Management (CIKM)*, 2004.

[25] Y. Yue and C. Burges. On using simultaneous perturbation stochastic approximation for IR measures, and the empirical optimality of LambdaRank. *NIPS Machine Learning for Web Search Workshop*, 2007.