A Machine Learning Approach for Improved BM25 Retrieval

Krysta M. Svore and Christopher J. C. Burges
Microsoft Research
One Microsoft Way
Redmond, WA 98052
{ksvore,cburges}@microsoft.com

Microsoft Research Technical Report MSR-TR-2009-92

July 30, 2009

Abstract

BM25 is one of the most widely used information retrieval functions because of its consistently high retrieval accuracy. Despite its widespread use, 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 when the document description is over single or multiple fields. 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.

1 Introduction

BM25 [16] is arguably one of the most important and widely used information retrieval functions. It has served as a strong baseline in the information retrieval community, in particular in the TREC Web track [5, 6]. Originally designed to

be computed over the body and title fields of a Web document, BM25 is a non-linear combination of three key document attributes: term frequency, document frequency, and document length.

Recent research suggests that using click information in ranking can significantly improve accuracy [1, 7, 24]. Since popularity fields are not generated by the author of the document, the question has been raised if the foundations behind BM25 are suitable for popularity fields [21]. We are particularly interested in the retrieval effectiveness of popularity fields. We empirically determine how well BM25 models content fields, such as the title, URL, and body fields, versus popularity fields, such as anchor text and query click. We demonstrate that BM25 works remarkably well for most content fields, but not for popularity fields. BM25F [17] is an extension of BM25 that prescribes how to combine more than one field in the document description, and correspondingly how to compute BM25 across the expanded document description. We determine the contribution of single and multiple field combinations to retrieval effectiveness and correspondingly evaluate the retrival capacity of BM25 on these fields.

A challenge to using BM25 and BM25F is the necessity of parameter tuning. Parameters control the contributions of term frequency, field length, and field weight. BM25F requires the tuning of $2K + 1$ parameters for a document description containing K fields. Tuning can be accomplished using a grid-search method, or by using gradient descent [21]. Each method has its drawbacks; grid-search can be time intensive and in fact prohibitively slow when the data collection is large, but can find a reasonable set of parameter values for a given target cost function. Gradient descent is much faster, but the method in [21] does not optimize the parameters directly for a target evaluation measure and finds parameters no better than the grid-search technique.

Recently, it has been shown that LambdaRank [3] is empirically optimal [8, 25] for Mean Average Precision, Mean Reciprocal Rank, and NDCG, and likely for other IR measures as well. We could extend the approach in [21] to use LambdaRank and thus optimize the parameters for a chosen IR measure, but the function is still restricted to the predefined BM25 and BM25F probabilistic models. The probabilistic model is not the only way to approach information retrieval; we consider a machine learning approach to develop a BM25-style model. A machine learning approach has been difficult previously due to the difficulty in obtaining large amounts of training data. A resulting challenge has been how to prevent a complex model with many parameters from overfitting the training data. However, by training LambdaRank [3] on the input attributes of BM25F and over a large data collection, we are able to build a rich, expressive retrieval model.

Our main interest is in developing an improved ranking model that uses the same input attributes as BM25; we therefore consider how to develop a retrieval

function that considers information across fields. Our model learns from the same attributes as BM25, namely term frequency, document frequency, and document length, and avoids parameter tuning. We learn the function directly from the data under consideration by training LambdaRank on the input attributes of BM25, while also optimizing directly for the IR evaluation measure of choice. We call our model LambdaBM25, since it uses the LambdaRank training method and the inputs of BM25. Our model is very easy to retrain periodically as attribute values change (for example, as anchor text and query click fields change over time), and can be used as a framework for learning other functions by substituting their input attributes. We believe our model offers value in the design of future information retreival systems.

Our primary contributions are threefold:

- We empirically determine the effectiveness of BM25 for different field types. 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.
- We develop a machine learning model, called LambdaBM25, that is based on the attributes of BM25 [16] and the training method of LambdaRank [3]. Our model is both fast and simple; it does not require any parameter tuning and is an extension of a state-of-the-art neural net ranking approach. It combines the input attributes of BM25 with the principles of machine learning and goes beyond the probabilistic model with a data-driven approach. In addition, LambdaBM25 optimizes for MAP, MRR, or NDCG, as well as potentially other IR measures [8, 25].
- We extend our empirical analysis to a document description over various field combinations. We confirm that BM25F [17] is better than a linear function of BM25 scores. We then extend our model, LambdaBM25, to document descriptions consisting of combinations of fields 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. BM25 [16] is based on a probabilistic information retrieval model [20] which incorporates attributes of documents, such as term frequencies, document frequencies, and document length. A generalized inverse document frequency model was recently developed that can also be incorporated into BM25 [13]. BM25 is one of the most widely used retrieval methods and serves as a standard baseline in the information retrieval community.

More recently, there has been interest in a simple retrieval function that can capture signals over multiple document fields. The development of this work began with Wilkinson [22], who evaluated various ways to weight scores that came from different fields of a document. Ogilvie and Callan [14] overview different field combination methods as well as propose several novel approaches with a complete evaluation. More recently, Robertson et al. proposed a simple method of combining attributes across multiple fields called BM25F [17].

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 parameters and length normalization parameters [18, 10]. Taylor et al. [21] proposed an approach based on gradient descent that significantly reduces the tuning time over using standard grid-search heuristics. Their technique does not, however, directly optimize for the evaluation measure or offer improved accuracy over a grid-search heuristic.

Recently, it has been shown that LambdaRank [3] is empirically optimal for several IR measures [8, 25], in particular for NDCG, Mean Average Precision, and Mean Reciprocal Rank. Our work is a combination of LambdaRank and BM25 attributes, and allows us to optimize directly for the IR measure under consideration.

Recent studies demonstrate the effectiveness of query click data for ranking [1, 7, 9, 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.

The remainder of the paper is laid out as follows. In the next section, we review the fields of a document. Section 4 provides an overview of BM25 and BM25F. In Section 5, we describe the evaluation measure NDCG [11] and the neural network ranking algorithm LambdaRank [3]. In Section 5.4, we discuss how to learn a BM25-like retrieval function over a large data collection. In Section 6, we describe our experiments and present our results. Finally, we conclude in Section 7 and discuss future directions of our of work.

3 Document Fields

A Web document is composed of several *fields* of information, in particular the title of the page, the URL, the body text, the anchor text, and queries that lead to a click on the page. The field may either be written by the owner of the page, as in the case of the body text, the URL, and the title of the document, or by other authors, as in

the case of anchor text and query click information. We call the former sources of information *content* fields and the latter sources of information *popularity* fields.

The *document description* is a concatenation of the available fields of information. The document desription may be restricted to certain fields, for example, if no click information is available or if a retrieval system does not crawl anchor text. To study the effects of various fields, we can restrict the document description to particular subsets of fields, or to a single field. Such studies can be enlightening and may lead to further improvements in information retrieval functions.

All field information is preprocessed by removing punctuation, converting to lowercase, and removing html markup. A query q is composed of terms; we consider queries which contain at most 10 terms. A document description is decomposable into smaller units, for example, characters, terms, or phrases; we follow the traditional decomposition into terms. The document frequency for term t is the number of documents in the collection that contain term t in their document descriptions. Note that the document description may be over one or several fields. The term frequency is calculated per term and per field by counting the number of occurrences of term t in field F of the document under consideration. We measure the length of a field by counting the number of terms in the field. In this section, we review the fields of a Web document.

3.1 Content Fields

The content fields of a document include the body text, the document's title, and the URL text. The body field consists of the html content of the page. It includes outgoing link information, image text, navigation bars, and so on, in addition to the core content. The title field contains the title of the document, indicated by the author through html ¡TITLE¿ tags. For example, for the site *http://webmessenger.msn.com*, the title field is *msn web messenger*. The URL field contains the text of the page's web address, after word breaking. For example, for the site *http://webmessenger.msn.com*, the URL field is *web messenger msn*. The body field is typically significantly longer than the URL and title fields.

3.2 Popularity Fields

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. Popularity fields can be highly repetitive for common pages, and can have a short length for lesser-known (tail) pages.

(msn web messenger, 1802)
(webmessenger, 1278)
(web messenger, 526)
(msn web, 176)
(access messenger via the web, 95)
(web msn, 78)
(web msn messenger, 65)
(msn messenger, 40)
(msn, 37)
(webmsn, 26)
(here, 8)
(msn webmessenger, 7)
(this, 5)
...

Figure 1: Extract of the anchor text field for the site *http://webmessenger.msn.com*.

3.2.1 Anchor Text Field

The anchor text field is composed of the text of all incoming links to the page. Anchor text is supposed to indicate the trustworthiness of the document; if people link to a page, it signals that those people trust its content. Figure 1 lists an extract of the anchor text field for the site *http://webmessenger.msn.com*. We compress the field by listing an anchor text string followed by the number of incoming links with that string. Note that this representation will preserve term ordering. Our representation of the anchor text field equally weights all incoming links, regardless of their parent page. In future research, it may be suitable to weight the text of an incoming link by the importance of the page it comes from, for example by its PageRank score [2].

The anchor text field can be highly repetitive. Most Web pages tend to have large numbers of incoming links containing identical terms. The field may be repetitive if there are incoming links with repetitive text, or the field may be elaborative if the text of incoming links is diverse. For example, in Figure 1, the first two examples of anchor text contribute a large number of repetitive terms to the anchor text field. Ideally, this behavior — repetitive versus elaborative — should be learned automatically from the document collection (see Sections 4 and 5.4).

3.2.2 Query Click Field

Another source of information for document retrieval is query click information. Query click information for a document, aggregated across many users, signals relevance for a given query. It shares the property with anchor text that the queries are not authored by the document's owner.

We follow [1, 9] and build the query click field from query session data. For details on the query click field, we refer the reader to [9]. We briefly review the field in this section. Our query click data consists of query sessions extracted from one year of a commercial search engine's query log files. A query session consists of a user-issued query and a ranked list of 10 documents, each of which may or may not be clicked by the user. A query session can be represented by a triplet (q, r, c) [12], where q is the query, r is the ranking of documents, and c is the set of documents the user clicked on.

In [9], the query click field 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 a score assigned to that query. $Score(d, q)$ could be the number of times the document was clicked on for that query, but it is important to also consider the number of times the page has been shown to the user and the position in the ranked list at which the page was shown.

The score in [9] represents the importance of the query q in describing the relevance of document d that does not consider position, but does consider the number of times the document has been shown to users. The score can be derived from raw click data as

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

where $C(d, q)$ is the number of times d is shown to the user when q is issued, also called the number of impressions, $C(d, q, click)$ 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. Since the last clicked document for a query is a good indicator of user satisfaction, the score is increased in proportion to β by the last click count. Figure 2 shows an extract of the query click field for the site *http://webmessenger.msn.com*, extracted from [9] for completeness. The term frequency of term t for the query click field is calculated as

$$\sum_{p|t \in p} Score(d, q), \tag{2}$$

where p is the set of query-score pairs.

(msn web, 0.6675749)
(webmessenger, 0.6621253)
(msn online, 0.6403270)
(windows web messenger, 0.6321526)
(talking to friends on msn, 0.6130790)
(school msn, 0.5994550)
(msn anywhere, 0.5667575)
(web message msn com, 0.5476839)
(msn messager, 0.5313351)
(hotmail web chat, 0.5231608)
(messenger web version, 0.5013624)
(browser based messenger, 0.3814714)
(im messenger sign in, 0.2997275)
(msn web browser download, 0.0926431)
(install msn toolbar, 0.0027248)
...

Figure 2: Extract of the query click field for the site *http://webmessenger.msn.com* [9].

4 BM25

In this section, we briefly review previous work on BM25, but refer the reader to [16, 20] for a complete description. BM25 [16, 20] stems from the 2-Poisson probabilistic model of information retrieval; the task is to answer "What is the probability that document d is relevant to query q?". The document d is constrained to the document description, which may be over one or several fields, as described in Section 3.

The classic retrieval function BM25 is a function of several field attributes: term frequencies, document frequencies, and the field length. Although the document description could be over several fields, BM25 traditionally has considered a document description restricted to a single field, or at most two fields, body and title. However, additional fields can provide different signals of relevance and help improve ranking accuracy. BM25F [17] is an extension of the BM25 function to a document description over multiple fields. A key property of this function is that it is nonlinear. Since BM25F reduces to BM25 when calculated over a single field, we will refer to both functions as BM25$_F$, where F is a specification of the fields contained in the document description.

BM25$_F$ is computed as follows for document d, with a document description

over fields F, and query q:

$$S = \sum_{t \in q} TF_t * I_t. \tag{3}$$

The sum is over all terms t in query q. I_t is the Robertson-Sparck-Jones form of inverse document frequency of term t and is calculated as

$$I_t = \log \frac{N - df + 0.5}{df + 0.5}, \tag{4}$$

where N is the number of documents in the collection, df is the document frequency of term t. Note that the document frequency is calculated across the entire document description. In our experiments, for simplicity, we calculate document frequency over the body field for all document frequency attributes[1].

TF_t is a simple term frequency saturation formula that limits the impact of observing a term multiple times in a field. It is defined as

$$TF_t = \frac{f}{k + f}, \tag{5}$$

where f is calculated as

$$f = \sum_F \frac{w_F * tf_F}{\beta_F}. \tag{6}$$

tf_F is the term frequency attribute of term t in field F, k is the saturation parameter that controls the nonlinearity of TF_t, β_F is a function of field length, defined below, and w_F is a tuned field weight parameter. TF_t satisfies three key properties: (1) When $tf_F = 0$, then $TF_t = 0$, (2) the function increases monotonically with tf_F and (3) it has an asymptotic limit.

The parameter k is used to tune the saturation of term frequency. If $k = 0$, the function reduces to 1 and we score the query document pair according to the presence of the term across the collection only. If k is large, the function is nearly linear in tf_F. Small k values are typical, say $1 - 2$, demonstrating that tf_F is highly nonlinear; after only a few occurrences of the term, the impact of additional occurrences is minimal.

The 2-Poisson model makes sense only when documents are of equal length, so BM25$_F$ includes a component to account for varying field lengths. Two documents relevant to the same topic may be different lengths because of wordiness attributable to either repetition or elaboration. The BM25$_F$ formula assumes wordiness is only attributable to repetition. The field length component is defined as

$$\beta_F = (1 - b_F) + b_F(\ell_F / avg\ell_F), \tag{7}$$

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

where b_F is the length tuning parameter, ℓ_F is the length of the field, and $avg\ell_F$ is the average length of the field in the document collection. b_F is a tuning constant between 0 and 1. If $b_F = 1$, then simple normalization is used, which is meant to correct for verbosity. If b_F is small, it reduces the effect of normalization.

The instantiation of BM25$_F$ requires parameter tuning and setting. 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 a simple grid-search technique or by using a gradient-descent method [21]. However, since the parameters should be tuned on a large dataset, tuning can be time intensive and potentially prohibitively slow. In our experiments (see Section 6), we tuned the parameters of BM25$_F$ using grid search over 10K queries. We note that the grid search for various field combinations for $K > 3$ took over 2 weeks to complete.

5 Learning a BM25-style Function

In this section, we describe our simple machine learning ranking model that uses the input attributes of BM25$_F$ and the training method of LambdaRank. Our approach is general and may be applied to other retrieval functions. It overcomes the obstacle of parameter tuning and is completely data driven. We begin by reviewing previous work: the target evaluation measure NDCG [11] and the training algorithm LambdaRank [3].

5.1 NDCG

We choose to evaluate using NDCG, which has been shown to be a good measure for relevance of web documents to a query. Normalized Discounted Cumulative Gain (NDCG) [11] is a widely used measure for search metrics. It operates on multilevel relevance labels. We assume in our work that relevance is measured on a 5-level scale. NDCG for a given query q is defined as follows:

$$\text{NDCG@}L_q = \frac{100}{Z} \sum_{r=1}^{L} \frac{2^{l(r)} - 1}{\log(1+r)} \tag{8}$$

where $l(r) \in \{0, \ldots, 4\}$ 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@$L_q = 100$. Mean NDCG@L is the normalized sum over all queries: $\frac{1}{N} \sum_{q=1}^{N} \text{NDCG@}L_q$. NDCG is particularly well-suited for Web search applications since it accounts for multilevel relevance labels and the truncation level can be set to model user behavior. In our studies, we

consider mean NDCG@$1, 3, 10$. For brevity, we write NDCG@$1, 3, 10$. DCG is simply NDCG (Eqn 8) without the $1/Z$ normalization factor.

5.2 LambdaRank

In the next two subsections, we review a state-of-the-art ranking algorithm called LambdaRank [3] that optimizes for IR measures. For complete details, we refer the reader to [3]. LambdaRank is both a list-based and a pair-based neural network learning algorithm; it is trained on pairs of documents per query, where documents in a pair have different relevance labels. It is an extension of RankNet [4], another pair-based ranking algorithm whose cost function is a sigmoid followed by a pair-based cross-entropy cost.

In most machine learning tasks, a target evaluation measure is used to evaluate the accuracy of the model at test time, and an optimization measure, generally a smooth approximation to the target measure, is used to train the system. Ideally, the optimization measure matches the target measure, but typical IR target costs (e.g. MAP, MRR, mean NDCG, etc.) are either flat or non-differentiable everywhere and require sorting by model score, which itself is a non-differentiable operation. Hence, direct optimization of the target measure is quite challenging. LambdaRank [3] 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.

LambdaRank provides a significant speed-up over RankNet as well as a new method for directly optimizing a cost function using λ-gradients. In the next section, we describe the λ-gradient for NDCG, although the gradient definition is general and can work with any target evaluation measure.

5.3 λ-Gradient for Mean NDCG

A LambdaRank gradient, λ_j, is defined to be a smooth approximation to the gradient of a target evaluation measure with respect to the score of the document at rank position j. λ-gradients have a physical interpretation; documents are represented by point masses and λ-gradients are forces on those point masses [3]. On two documents in a pair in a query, the λ-gradients are equal and opposite, where a positive λ-gradient indicates a push toward the top of the list, and a negative λ-gradient indicates a push toward the bottom of the list. With a suitably defined λ-gradient, the gradient of any target evaluation measure can be smoothly approximated for a given document.

In [3], several alternatives for λ-gradients are given, and the best λ-gradient definition is chosen according to accuracy on validation data. The best λ-gradient found in [3] is a combination of the derivative of the RankNet cost [4] scaled by the NDCG@L_q gain from swapping two documents i and j with differing labels for a query q. We drop q below for brevity.

The RankNet cost is a pairwise cross-entropy cost applied to the logistic function on the difference of the model scores. Assume document i has score s_i and relevance label l_i, document j has score s_j and relevance label l_j, and $o_{ij} \equiv s_i - s_j$ is the score difference, then the RankNet cost can be written as follows:

$$C_{ij} \equiv C(o_{ij}) = -S_{ij} o_{ij} + \log(1 + e^{S_{ij} o_{ij}}), \tag{9}$$

where

$$S_{ij} = \begin{cases} +1 & \text{if } l_i > l_j \\ -1 & \text{if } l_i < l_j \end{cases} \tag{10}$$

The derivative of the RankNet cost according to score difference is

$$\delta C_{ij} / \delta o_{ij} = \delta C_{ij} / \delta s_i = -S_{ij} / (1 + e^{S_{ij} o_{ij}}). \tag{11}$$

The λ-gradient can now be expressed as

$$\lambda_{ij} \equiv S_{ij} \left| \Delta\text{NDCG} \frac{\delta C_{ij}}{\delta o_{ij}} \right| \tag{12}$$

$$= S_{ij} \left| N(2^{l_i} - 2^{l_j}) \left(\frac{1}{\log(1 + r_i)} - \frac{1}{\log(1 + r_j)} \right) \left(\frac{1}{1 + e^{S_{ij} o_{ij}}} \right) \right|$$

where N is the reciprocal of the maximum DCG for the query and r_i and r_j are the rank positions of documents i and j, respectively. Note that the sign S_{ij} only depends on the labels of documents i and j and not on their rank positions. In addition, if $l_i > l_j$, then document i is more relevant than document j and document i must move up the ranked list to reduce the cost, so $S_{ij} = 1$ and the λ-gradient for document i is positive.

The λ-gradient for a single document is computed by marginalizing over the pairwise λ-gradients,

$$\lambda_i = \sum_{j \in P} \lambda_{ij}, \tag{13}$$

where the sum is over all pairs P for query q which contain document i (see [3] for details).

5.4 LambdaBM25

Retrieval can be treated as a ranking process, where the ranking model ranks documents in order of decreasing relevance to the query q. In the probabilistic IR model BM25, documents are ranked by probability of relevance to the query. However, there are several challenges to using BM25, including the requirement of parameter tuning, the inability to directly optimize for an IR measure, and the restrictions of the underlying probabilistic model. In this section we directly address these challenges by introducing a new machine learning approach to BM25-like retrieval. Our model, called LambdaBM25, is trained using LambdaRank due to its flexibility, ease of training, and state-of-the-art ranking accuracy. It employs the NDCG λ-gradient previously described and learns a function of the BM25_F attributes directly from the data collection.

As mentioned previously, BM25 can be prohibitively expensive when trained on a document description over many fields. With the growing use of anchor text and click information, and potentially other metadata, training parameters for BM25_F can be costly. LambdaBM25 does not require parameter tuning since the function is learned directly from the train collection. In addition, LambdaBM25 can be extended to optimize for several IR measures, since LambdaRank has recently been shown to be empricially optimal for NDCG and other IR measures [8, 25].

A basic assumption behind the BM25 formula is that two documents about the same topic may be different lengths because one is more verbose. However, it may not be true that verbosity is the only reason a document is longer. Verbosity could imply elaboration or that the document covers multiple topics and not mere wordiness, in which case it may be appropriate to assign a longer document a larger BM25 score, whereas typically it would be assigned a smaller score. The BM25 formula cannot account for such differences, while LambdaBM25 has the flexibility to learn from the data if the documents tend to be verbose or elaborative. In addition, wordiness may be common among some fields and rare among others. For example, a title or URL field is succinct, while anchor text and query click fields are verbose due to repeatability. Our method learns these differences through neural net training and can apply different functions to the fields in the document description.

Our model has the additional advantage that it does not require that the attributes be statistically independent, as in [20]. LambdaBM25 learns relationships among the attributes and the fields from the data collection through LambdaRank training that may not be apparent otherwise. Our machine learning approach to an improved BM25-style function is trained over a very large data collection so that our model is effective, robust, and avoids overfitting.

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 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 optimize for NDCG and use the λ-gradient as previously described. 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 with varying numbers of hidden nodes. Since neural network learning improves when the data is normalized, we apply several transformations to the input attributes to achieve zero mean, unit variance across the feature values. Results are discussed in Section 6.

6 Experiments

We perform extensive experiments to determine the effectiveness of BM25 on single fields and multiple-field combinations and to determine the most important fields in a document. We then compare our method, LambdaBM25, to BM25 and evaluate the techniques on a very large train and test collection. Our goal is to have our nonlinear (two-layer) LambdaBM25 model demonstrate improved accuracy over BM25.

6.1 The Data and Evaluation Measure

We evaluate our method on a real-world Web-scale data collection. The data contains queries sampled from query log files of a commercial search engine and corresponding URLs. All queries are English queries and can contain up to 10 query terms. Our data collection includes anchor text, title, URL, body, and query click fields. We perform stopword removal and some stemming on queries. Field information is preprocessed as previously described.

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. The features consist of the term frequencies for terms in positions 1–10, the document frequencies for terms in positions 1–10, and field lengths for all fields under consideration. Each query-URL pair also has a relevance label. The label is human generated and is on a 5-level relevance scale, 0 to 4, with 4 meaning document d is the most relevant

Table 1: Parameters learned using grid search on the validation set for single-field BM25$_F$.

Field F	k_F	b_F	$avg\ell_F$
Title (T)	3.000	0.400	7
URL (U)	1.135	0.331	6
Body (B)	1.000	0.500	1815
Anchor (A)	0.910	0.008	167
Click (C)	101.540	0.504	39

to query q and 0 meaning d is not relevant to q.

We evaluate model performance using mean NDCG. We report NDCG scores at truncation levels 1, 3, and 10. We also perform a signficance test, i.e., a t-test with a significance level of 0.05. A significant difference should be read as significant at the 95% level. Statistical significance between pairs of models is indicated in bold.

6.2 Effectiveness of Single Fields

We first seek to determine which single field is the most effective in terms of ranking relevant documents using BM25$_F$. The parameters of BM25$_F$, where here F is the single field in the document description, are tuned to optimize NDCG@1 on our validation set using a 2-D grid search over the saturation parameter k_F and the length normalization parameter b_F, for each field F. We follow the grid search method outlined in [21], except we consider 1000 epochs or convergence of NDCG@1 as the stopping criterion. It was prohibitively slow to tune the parameters on the training set due to its size. Table 1 lists the parameters found for each individual field. We also tried an approach similar to the gradient-based approach in [21] and found results to be almost identical.

In Table 2, we report results for BM25$_F$ on a document description restricted to a single content or popularity field. The three content fields, Title (T), URL (U), and Body (B), are equally effective in terms of NDCG ranking accuracy on our test set. At truncation level 10, the body field yields significantly better ranking accuracy. The URL field appears to be the least reliable for retrieval in terms of accuracy across the three truncation levels.

For popularity fields, retrieval using only the anchor text field (A) yields improved NDCG scores over retrieval using a single content field. However, BM25$_C$ over the query click field yields almost a 7 point NDCG gain at truncation level 1 and a 4 point NDCG gain at truncation level 3 over BM25$_A$. Certainly, if restricted to a single field, the query click field achieves the highest NDCG accuracy.

We next seek to compare BM25$_F$ to single-layer LambdaBM25$_F$ on single

Table 2: Accuracy results on the test set for BM25_F for single fields.

Model	NDCG@1	NDCG@3	NDCG@10
BM25_T	24.50	27.23	33.32
BM25_U	24.96	27.24	32.77
BM25_B	24.35	27.92	35.07
BM25_A	33.50	32.53	33.37
BM25_C	40.07	36.62	35.89

Table 3: Accuracy results on the test set for 1-layer LambdaBM25_F for single fields. 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.

Model	NDCG@1	NDCG@3	NDCG@10
LambdaBM25_T	*20.79*	*24.93*	*32.51*
LambdaBM25_U	*22.96*	*26.38*	33.17
LambdaBM25_B	*18.03*	*21.93*	*30.60*
LambdaBM25_A	(33.83)	**33.11**	**34.73**
LambdaBM25_C	*39.34*	(36.50)	(35.96)

field document descriptions. Since BM25_F is a highly nonlinear function, we expect it to outperform a simple linear combination of input attributes, in particular for the content fields, for which BM25_F was originally developed. 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 [20]. We train single-layer LambdaBM25_F models by choosing the best training epoch and learning rate based on the validation data. We found a learning rate of 10^{-5} and 500 epochs to be reasonable settings for all fields.

Table 3 contains results for single-layer LambdaBM25_F. At each truncation level, our results indicate that for each content field, BM25_F significantly outperforms our learned linear function at each truncation level, with the exception LambdaBM25_U, which performs similarly to BM25_U at truncation level 10. For content fields, we conclude that BM25_F is significantly better than a linear combination of input attributes. We anticipated such a result since BM25_F was explicitly designed for improved accuracy over a linear term frequency function when using content fields.

In the case of popularity fields, the results indicate that our single-layer LambdaBM25_F model performs similar or better than BM25_F. For the anchor text field, we find

Table 4: Number of hidden nodes found on the validation data for single field, two-layer LambdaBM25$_F$.

Field F	Hidden nodes
T	10
U	15
B	15
A	15
C	5

Table 5: Accuracy results on the test set for 2-layer LambdaBM25$_F$ for single fields. 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.

Model	NDCG@1	NDCG@3	NDCG@10
LambdaBM25$_T$	(24.31)	(27.38)	**33.86**
LambdaBM25$_U$	*23.69*	*26.70*	**33.21**
LambdaBM25$_B$	**27.53**	**30.49**	**37.03**
LambdaBM25$_A$	**36.33**	**34.68**	**35.33**
LambdaBM25$_C$	**41.61**	**38.01**	**37.19**

that BM25$_A$ performs significantly worse at truncation levels 3 and 10 than our learned linear function LambdaBM25$_A$. Similarly, for the query click field, we find that BM25$_C$ performs similarly to our learned linear function LambdaBM25$_C$. Such results were hypothesized in [21], and 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 seek to determine if our nonlinear LambdaBM25$_F$ model can outperform BM25$_F$. We train a two-layer neural net with 5, 10, and 15 hidden nodes, for various learning rates. We choose the best net according to the validation set. We found a learning rate of 10^{-5} and 500 epochs to consistently perform well. Table 4 lists the number of hidden nodes and training parameters used for each single-field nonlinear model.

Table 5 reports the results of BM25$_F$ versus our learned two-layer LambdaBM25$_F$ model. For the Title field, BM25$_T$ performs almost identically to LambdaBM25$_T$. For the URL field, BM25$_U$ performs slightly better at most truncation levels than LambdaBM25$_U$. We conclude that BM25 models these two content fields very

well. However, for the Body field, we find that LambdaBM25$_B$ outperforms significantly BM25$_B$ across all truncation levels. We hypothesize that BM25$_F$ models short, succinct, non-repeatable fields well, but fails to model longer fields with similar accuracy. Both the Title and URL fields are reasonably short, while the Body field on average is around 300 times longer. 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.

For popularity fields, we find that two-layer LambdaBM25$_F$ consistently outperforms BM25$_F$, with statistical significance, which further confirms that the BM25$_F$ function was not designed for popularity fields. LambdaBM25$_F$ is able to exploit found relationships in the training data that are restricted in the BM25$_F$ model.

6.3 Effectiveness of Multiple Fields

For a document description over a single field, BM25$_F$ exhibits reasonable accuracy for content fields, while LambdaBM25$_F$ exhibits superior accuracy for popularity fields. We have also seen that with query click information alone, we can achieve substantial retrieval accuracy gains. In this section, we perform experiments to examine retrieval effectiveness when the document description contains multiple fields. We find that our learned nonlinear method, LambdaBM25$_F$, outperforms BM25$_F$ when F is a document description over multiple fields. We also verify that a nonlinear combination of multiple fields is required for the best retrieval accuracy.

The parameters of BM25$_F$ are tuned using a $2K$-dimensional grid search, where K is the number of fields in the document description. We consider several combinations of fields; the combinations and their parameters are listed in Table 6. Note that the parameter k can be absorbed into the field weights w_F (see Eqs 5–6). Thus we assume $k = 1$ and learn $2K$ parameters instead of $2K + 1$.

All field weights, with the exception of the query click field weight w_c, are between 0 and 20. In all field combinations, the body field consistently receives the lowest field weight w_B. When the document description is over all fields (final row of the table), the query click field receives a weight 1000 times more than the anchor text or body fields. We can conclude that the query click field is the most important field in the document description.

We first seek to determine the most effective combination of fields to include in the document description for BM25$_F$. Table 7 lists 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

Table 6: BM25$_F$ parameters learned on the validation set for various field combinations. In all cases, $k = 1$.

Fields F	w_T	b_T	w_B	b_B	w_U	b_U	w_A	b_A	w_C	b_C
T, B	9.8000	0.5406	0.0044	0.420	-	-	-	-	-	-
T, B, U	0.6454	0.4009	0.0085	0.196	1.6219	0.950	-	-	-	-
T, B, U, A	2.2415	0.6440	0.0490	0.685	5.4024	0.989	0.3202	0.0083	-	-
T, B, U, C	1.5811	0.9250	0.0290	0.851	6.7961	0.979	-	-	2.9265	0.390
T, B, U, A, C	9.5327	0.7990	0.0944	0.430	18.7944	0.890	0.0935	0.1993	93.8318	0.528

Table 7: Accuracy results on the test set for BM25$_F$ for multiple fields.

Model	Fields F	NDCG@1	NDCG@3	NDCG@10
BM25$_F$	T, B	27.84	30.81	36.98
BM25$_F$	U, T, B	30.81	33.30	39.53
BM25$_F$	A, U, T, B	38.66	38.83	43.42
BM25$_F$	C, U, T, B	45.29	43.37	46.83
BM25$_F$	C, A, U, T, B	45.41	43.53	46.88

Table 8: Accuracy results on the test set for 1-layer LambdaBM25$_F$ for multiple fields. 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.

Model	Fields F	NDCG@1	NDCG@3	NDCG@10
LambdaBM25$_F$	T, B	*25.42*	*28.81*	*35.80*
LambdaBM25$_F$	U, T, B	*29.28*	*32.08*	*38.75*
LambdaBM25$_F$	A, U, T, B	(38.91)	(38.84)	*42.81*
LambdaBM25$_F$	C, U, T, B	*43.34*	*41.70*	*45.04*
LambdaBM25$_F$	C, A, U, T, B	*44.60*	*42.33*	*45.44*

that include the query click field. Note that using multiple fields outperforms using only BM25$_C$. Even using the anchor text field in conjunction with all content fields cannot match the accuracy of BM25$_C$. The addition of anchor text to the C,U,T,B combination in fact yields very little improvement in accuracy, without statistical significance. The anchor text field is, however, important when query click information is not available, as we can see by the significant accuracy improvement between the U,T,B and A,U,T,B field combinations.

We next 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 listed in Table 6. For each model, we find a learning rate of 10^{-5} performs best on our validation data. Table 8 lists the results of our learned linear function LambdaBM25$_F$. In all cases, we find that BM25$_F$ performs as well or better than single-layer LambdaBM25$_F$; our results confirm that the motivation for BM25F given in [17] is accurate that a linear combination of fields is insufficient for good retrieval accuracy.

Finally, we seek to determine if our two-layer LambdaBM25$_F$ model learns a better BM25-style retrieval function than BM25$_F$. We train two-layer LambdaBM25$_F$ models on the field combinations listed in Table 6. We find that a learning rate of

Table 9: Accuracy results on the test set for 2-layer LambdaBM25$_F$ for multiple fields. Bold indicates statistical significance over the corresponding BM25$_F$ model.

Model	Fields F	NDCG@1	NDCG@3	NDCG@10
LambdaBM25$_F$	T, B	**29.61**	**32.49**	**38.93**
LambdaBM25$_F$	U, T, B	**34.26**	**37.03**	**43.05**
LambdaBM25$_F$	A, U, T, B	**43.70**	**42.58**	**46.21**
LambdaBM25$_F$	C, U, T, B	**49.70**	**46.58**	**49.14**
LambdaBM25$_F$	C, A, U, T, B	**50.33**	**47.14**	**49.47**

10^{-5} and 15 hidden nodes performs well for all field combinations on our validation data. Table 9 reports results of BM25$_F$ versus two-layer LambdaBM25$_F$ for various field combinations. For every field combination, LambdaBM25$_F$ achieves gains with statistical significance over the corresponding BM25$_F$ model. As expected, we see smaller gains between LambdaBM25$_{T,B}$ and BM25$_{T,B}$ since BM25$_F$ models title and body fields very well. For combinations that include anchor text and query click fields on the otherhand, we see very substantial gains over BM25$_F$ of around 5 points NDCG@1 and 3 points NDCG@10. Note that even 0.5 points NDCG gain is substantial, in particular for truncation level 1.

We would like to highlight that for both BM25$_F$ and two-layer LambdaBM25$_F$ models, the gains achieved when new fields are added to the document description are consistent. In Tables 7 and 9, the inclusion of the query click field in the document description yields the highest accuracy. In addition, smaller gains are achieved by adding the anchor text field to the document description. Ordering by accuracy, the multiple-field combinations are in the same order for BM25$_F$ and LambdaBM25$_F$.

7 Conclusions and Future Work

We have extensively studied the contributions of various document fields to information retrieval accuracy. We find that query click information is the most effective field, while the URL field is the least effective field. A document description containing all fields yields the best retrieval accuracy. We also study when BM25 outperforms a linear combination of input attributes. BM25 performs remarkably well on single content fields, but on single popularity fields, BM25 achieves a retrieval accuracy comparable to a linear function of input attributes. For document descriptions containing multiple fields, we verify that a nonlinear combination of field attributes attains significantly better accuracy than a linear combination of

field attributes.

Our main contribution is a new information retrieval model trained using LambdaRank and the input attributes of BM25. LambdaBM25$_F$ 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 of parameter tuning, yielding a significantly faster approach. Our model is general and can potentially act as a framework for modelling other retrieval functions.

There are several future directions of this work. First, we would like to perform more extensive studies to determine the importance of attributes in our model. Since LambdaBM25 is a neural network, it is difficult to determine the actual relationship learned between attributes. However, by using a decision tree learner, such as LambdaMART [23], we can decipher the trees to determine the ranked list of important features in our model. Currently, our preliminary results using LambdaMART to learn a BM25-style function indicate that term frequency attributes are significantly more important to the model than document frequency attributes. The most important features are the term frequencies of the first two terms of the query in the query click field and the title field. In addition, the field lengths of the body field and the query click field are the most important field length attributes.

Second, we would 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, LambdaBM25 could be used as a single feature in recent TREC retrieval systems [6, 5].

Finally, we plan to expand our model to learn proximity relationships. Recent work on incorporating proximity information into BM25 has focused on bigram frequencies [15] or frequencies of terms in spans [19]. In both cases, it has been unclear how to combine n-gram document frequency information with n-gram term frequency information. In addition, a challenge has been how to extend BM25 to account for relationships between a query term appearing as a unigram or with another query term as a bigram. We plan to examine the effect of bigram and trigram frequency attributes on our model and determine if incorporating such features can learn a better function than, for example, the proximity BM25 model given in [15, 19]. In the presence of proximity field attributes, we expect different field combinations to yield the highest retrieval accuracy. LambdaBM25 has the advantage of learning the relationship directly from the training collection and requires no tuning of the function. With our approach, we can learn the dependencies between document and term frequency directly.

8 Acknowledgments

We would like to thank Susan Dumais for providing many useful discussions and insights. We also thank Bhuvan Middha for invaluable discussions regarding the parameter tuning of BM25 and BM25F.

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] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Seventh International Conference on the World Wide Web (WWW)*, pages 107–117, 1998.

[3] C.J.C. Burges, R. Ragno, and Q.V. 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.

[4] C.J.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.

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

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

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[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.J.C. Burges, K.M. 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.J.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.