# Improvements to BM25 and Language Models Examined

Andrew Trotman
Department of Computer Science
University of Otago
Dunedin, New Zealand

Antti Puurula
Department of Computer Science
University of Waikato
Hamilton, New Zealand

Blake Burgess
Department of Computer Science
University of Otago
Dunedin, New Zealand

## ABSTRACT

Recent work on search engine ranking functions report improvements on BM25 and Language Models with Dirichlet Smoothing. In this investigation 9 recent ranking functions (BM25, BM25+, BM25T, BM25-adpt, BM25L, $TF_{l\circ\delta\circ p}\times ID$, LM-DS, LM-PYP, and LM-PYP-TFIDF) are compared by training on the INEX 2009 Wikipedia collection and testing on INEX 2010 and 9 TREC collections. We find that once trained (using particle swarm optimization) there is very little difference in performance between these functions, that relevance feedback is effective, that stemming is effective, and that it remains unclear which function is best overall.

## Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *Search process*.

## General Terms

Algorithms, Measurement, Performance, Experimentation.

## Keywords

Relevance Ranking, Document Retrieval, Procrastination

## 1. INTRODUCTION

Substantial precision improvements have been made in search engines since their inception. Initially these improvements were hard to quantify because each investigator used their own document collection. NIST addressed many of the early problems with the introduction of TREC [4]. For the TREC *ad hoc* experiments all participants used the same documents and queries. Participants were asked to submit, for each query, a ranked list of the most relevant documents, a *run*. NIST pooled the top *n* results from each run and asked information experts to assess which documents were relevant to which queries – a process known as *pooled assessment*. The assessments were used to measure, using a standard metric, the performance of each run. Software to produce the metric from a run and the assessments were made publicly available, as were the documents and queries. TREC made it possible for researchers to obtain all the necessary components to conduct a repeatable laboratory experiment in relevance ranking, it was soon followed by other evaluation forums including NTCIR [6], INEX [3], and FIRE [14]. These collections continue to be used to demonstrate improvements.

Armstrong *et al*. [1, 2] show that, although research continued, there was no evidence to suggest that any improvements in ranking had been made since the mid-1990s, the point at which the BM25 [23] ranking function was introduced. Trotman & Keeler [28] suggests that this is because BM25 performs at near human levels on the TREC collections.

It is reasonable to believe that commercial search engines continue to improve in precision. But they are tackling a different problem. For TREC *ad hoc* retrieval the practitioner is given the documents and the queries, from which they are expected to produce a run. Contrariwise, a commercial search engine has a query log, a click log, and user profiles. Commercial search engines also have a very different query profile from that at TREC. TREC queries are *unique* and *representative*, whereas commercial search engines see the same queries repeated and are able to optimize on a query-by-query basis.

Both the commercial and academic search engines see new queries and must produce a ranked results list. That list might become the input to a pipeline that further re-orders the results – a process known as result *re-ranking*. A typical re-ranking might be produced by a learning-to-rank algorithm. In this investigation we are concerned with the first stage of this pipeline. Given the documents and the query, produce the initial ranked list of results; the TREC run.

Many of the runs seen at TREC are not simply the result of a simple ranking function, but of a pipeline of other processes (such as stemming, stopping, field weighting, relevance feedback, and so on). We are not experimenting with this multitude of such possibilities as doing so exhaustively is prohibitive. We are primarily interested in choosing a default ranking function that can be expected to perform well everywhere and then allowing the practitioner to further tune that by adding (or not) extra filters to the pipeline. We are, however, interested in building on the results of Armstrong *et al*. That is, we ask:

*Has there been any improvement in ranking function precision?*

We examine this question by implementing and testing several recent ranking functions that claim improvements on BM25. We test each function, add relevance feedback, stemming, and stopping, and show that although improvements on BM25 appear to have been made, those improvements are small and it is unclear which ranking function is best overall.

## 2. EXPERIMENTAL CONDITIONS

The purpose of this investigation is to examine whether or not improvements in *ad hoc* retrieval have been made in recent years. As reproducibility is paramount, the experiments are conducted using resources from TREC and INEX.

Specifically, we train on the INEX 2009 Wikipedia collection and test on the INEX 2010 collection, as well as TREC 1 – TREC 8. The title of the topics were used as the queries, except for TREC 4 where only description fields are present and so they were used.

We implemented the functions using the ATIRE search engine [27], the source code of which is publicly available along with our extensions. The baseline for comparison was chosen as BM25.



*ADCS '14, November 27 - 28 2014, Melbourne, VIC, Australia*
*Copyright 2014 ACM 978-1-4503-3000-8/14/11…$15.00*
*http://dx.doi.org/10.1145/2682862.2682863*

Mean un-interpolated average precision (MAP) was used as the metric. This was measured to document 1000 in the results lists. It is common to cut the results list at some point earlier, but we cut late because we are looking for any evidence of improvements, not just evidence in the top few. We define MAP as the mean, over all queries in the query set, of the average precision of each query in the query set,

$$MAP = \frac{\sum_{q \in Q} AP_q}{Q} \tag{1}$$

where

$$AP_q = \frac{\sum_{n=1}^{L} \begin{cases} P_{qn}, & L_n \; relevant \\ 0, & otherwise \end{cases}}{N_{qr}} \tag{2}$$

where $N_{qr}$ is the number of known relevant documents for query $q$, $L$ is the length of the results list (at most 1000), $L_n$ is the result at position $n$ in the results list, $P_{qn}$, is the precision at that position (the number of relevant documents that have been found at point $n$, divided by $n$).

We are not making claims about a particular system, we are making claims about particular ranking functions within a particular system. As such a comparison to the best scores seen at TREC and INEX are immaterial – however we note that the Reference Run at INEX was produced using ATIRE BM25, s-stemming, but without feedback or stop words.

# 3. BM25

BM25 is often used as a baseline, and we do the same here. Problematically, many of the implementations of BM25 are different and comparisons are to BM25-like functions. We use the variant of BM25 as seen in ATIRE as our baseline.

## 3.1 ATIRE BM25

The ATIRE variant of BM25 was chosen to avoid negative numbers. The implementation is given by Trotman *et al*. [27] as:

$$rsv_q = \sum_{t \in q} \log\left(\frac{N}{df_t}\right) . \frac{(k_1 + 1) . tf_{td}}{k_1 . \left((1 - b + b . \left(\frac{L_d}{L_{avg}}\right)\right) + tf_{td}} \tag{3}$$

for a given query, $q$, the retrieval status value, $rsv_q$, is the sum of individual term, $t$, scores. $N$ is the number of documents in the collections, $df_t$ is the number of documents containing the term (the *document frequency*), $tf_{td}$ is the number of times term $t$ occurs in document $d$. $L_d$ is the length of the document (in terms) and $L_{avg}$ is the mean of the document lengths. There are two tuning parameters, $b$, and $k_1$.

This equation differs from the Robertson *et al*. equation [22] in several ways. The $k_3$ component of the Robertson *et al*. function accounts for terms that occur in the query more than one, whereas this function assumes each term in the query is unique. The $k_2$ parameter in the Robertson *et al*. function accounts for the query length and as they set $k_2$ to zero, so to have the authors of ATIRE.

The most striking difference in the ranking functions is the change to the IDF component. Robertson *et al*. use the Robertson-Sparck Jones IDF,

$$IDF_t = \log \frac{N - df_t + 0.5}{(df_t + 0.5)} \tag{4}$$

which produces negative scores when $df_t > N/2$ whereas ATIRE uses the Robertson-Walker IDF [8] $N/df_t$, which tends to zero as $df_t$ tends to $N$.

This change is intuitive. If the entire collection is ranked on $rsv_q$, then for a 1-term query containing a term that occurs in more than half the documents, the Robertson-Sparck Jones IDF ranks all documents not containing that term higher than those that do whereas the ATIRE function always considers documents containing the term to be more relevant than those that do not.

The ATIRE implementation of BM25 has proven to be effective in a number of scenarios, and by independent authors [18].

## 3.2 BM25L

Lv & Zhai [12] observe that the document length normalization of BM25 ($L_d/L_{avg}$) unfairly prefers shorter documents to longer ones. An observation made earlier by Singhal *et al.* [26], with respect to cosine ranking. Lv & Zhai address this problem in their BM25L function; their derivation is as follows:

Starting with a different BM25 that disallows negative values, again differing only in the IDF component:

$$rsv_q = \sum_{t \in q} \log\left(\frac{N + 1}{df_t + 0.5}\right) . \frac{(k_1 + 1) . tf_{td}}{k_1 . \left((1 - b) + b . \left(\frac{L_d}{L_{avg}}\right)\right) + tf_{td}} \tag{5}$$

They re-arrange to get

$$rsv_q = \sum_{t \in q} \log\left(\frac{N + 1}{df_t + 0.5}\right) . \frac{(k_1 + 1) . c_{td}}{k_1 + c_{td}} \tag{6}$$

where

$$c_{td} = \frac{tf_{td}}{1 - b + b . \left(\frac{L_d}{L_{avg}}\right)} \tag{7}$$

A rearrangement seen in other functions discussed in this section.

For BM25L, Lv & Zhai are interested in affecting this $c_{td}$ component to avoid over penalizing long documents. They do this by adding a positive constant, $\delta$, to it. This has the effect of shifting the function to better favor small numbers (i.e. large denominators, equivalently large $L_d$ values, or long documents).

Their final equation, BM25L, is given as:

$$rsv_q = \sum_{t \in q} \log\left(\frac{N + 1}{df_t + 0.5}\right) . \frac{(k_1 + 1) . (c_{td} + \delta)}{k_1 + (c_{td} + \delta)} \tag{8}$$

They show BM25L outperforming BM25 on several TREC collections including: GOV2, WT10G, WT2G, Robust04, and TREC-8. The best value for $\delta$ was 0.5 in all their experiments.

## 3.3 BM25+

Lv & Zhai [11] go on to observe that the penalization of long documents occurs not only in BM25 but other ranking functions. They propose a general solution to this, which is to lower-bound the contribution of a single term occurrence. That is, no matter how long the document, a single occurrence of a search term contributes at least a constant amount to the retrieval status value.

They do this differently from BM25L; they add $\delta$ to the $tf_{td}$ component before multiplying by IDF (which they also change).

$$rsv_q = \sum_{t \in q} \log\left(\frac{N + 1}{df_t}\right) . \left(\frac{(k_1 + 1) . tf_{td}}{k_1 . \left((1 - b) + b . \left(\frac{L_d}{L_{avg}}\right)\right) + tf_{td}} + \delta\right) \tag{9}$$

They use the TREC GOV2, WT100g, WT2G, and Robust04 collections and show that BM25+ outperforms BM25 in all cases. They suggest that a $\delta$ value of 1 is effective across collections.

## 3.4 BM25-adpt

In different work on BM25, Lv & Zhai [10] observe that a global $k_1$ applied to all query terms is likely to be less effective than a term-specific $k_1$, and seek to identify term-specific $k_1$ values directly from the index. This makes their ranking function transferable from collection to collection without re-training.

They apply information gain and divergence from randomness theory to the problem. Starting with the probability of seeing at least one occurrence given 0 or more occurrences and the query:

$$p(1|0,q) = \frac{df_r + 0.5}{N + 1} \tag{10}$$

they derive the probability of seeing one more occurrence as:

$$p(r+1|r,q) = \frac{df_{r+1} + 0.5}{df_r + 1} \tag{11}$$

from which the information gain at any point in the function can be computed as the change from $r$ to $r + 1$ occurrences, minus the initial probability:

$$G_q^r = log_2\left(\frac{df_{r+1} + 0.5}{df_r + 1}\right) - log_2\left(\frac{df_{tr} + 0.5}{N + 1}\right) \tag{12}$$

$df_r$ is more complex. Rather than using term frequency, $tf_{td}$, values, they define $df_r$ based on the result of the length normalized term frequency, equation (7). They do that by defining $df_r$ as:

$$df_r = \begin{cases} \left|D_{t|c_{td} \geq r-0.5}\right| & r > 1 \\ df_t & r = 1 \\ N & r = 0 \end{cases} \tag{13}$$

that is, for the base case of $r = 0$, the number of documents in the collection is used; when $r = 1$, the document frequency is used; in all other cases, $\left|D_{t|c_{td} \geq r-0.5}\right|$, the number of documents, $|D_t|$, containing the term, $t$, that have a length normalized occurrence count, $c_{td}$, greater than $r$ (once rounded).

To compute $k_1$, they align the information gain function to the BM25 score function and solve for $k_1$ giving the term specific $k_1'$.

$$k_1' = \arg\min_{k_1} \sum_{i=0}^{T} \left(\frac{G_q^r}{G_q^1} - \frac{(k_1 + 1).r}{k_1 + r}\right)^2 \tag{14}$$

They can compute $k_1'$ entirely from the index because all parameters are there – but they suggest pre-computing these values and storing them in the index, one per term.

Finally, $k_1'$ gets substituted into the term frequency component of BM25, and the IDF score replaced by $G_q^1$:

$$rsv_q = \sum_{t \in q} G_q^1 . \frac{(k_1' + 1).tf_{td}}{k_1' . \left((1 - b) + b . \left(\frac{L_d}{L_{avg}}\right)\right) + tf_{td}} \tag{15}$$

Lv & Zhai test their function on TREC-8, WT10g, and WT2g, and show that it out performs BM25.

## 3.5 BM25T

Lv & Zhai [13] later propose a log-logistic method of calculating term-specific $k_1$ parameters, again directly out of the index.

They first define the elite set of a term, $C_w$, as the set of all documents containing the term. Within this set they suggest that the length normalized term frequency contribution should be proportional to the proportion of documents with a higher length normalized term frequency contribution. They then use the log-moment estimation method to estimate the term specific $k_1$, $k_1'$ as:

$$k_1' = \arg\min_{k_1} \left(g_{k_1} - \frac{\sum_{D \in C_w} log(c_{td}) + 1)}{df_t}\right)^2 \tag{16}$$

where $c_{td}$ is defined in equation (7) and $g_{k_1}$ is defined as:

$$g_{k_1} = \begin{cases} \frac{k_1}{k_1 - 1} . \log(k_1) & if\ k_1 \neq 1 \\ 1 & otherwise \end{cases} \tag{17}$$

which they solve using the Newton-Raphson method. This $k_1'$ is then substituted for $k_1$ in equation (5) giving BM25T.

They suggest this method of estimating $k_1$ might be unstable when the document frequency is small (there isn't much information). To address this they introduce BM25C which uses the mean $k_1'$ over all terms seen in all queries (which is unknowable), and BM25Q which uses the mean $k_1'$ for all terms in the current query.

Tests on WT2G, WT10G, AP, and Robust04 show that these functions outperform BM25. However which is best is unclear. For this investigation we have experimented with BM25T

## 3.6 $TF_{l\circ\delta\circ p}$×IDF

Rousseau & Vazirgiannis [25] suggest that nonlinear gain from observing an additional occurrence of a term in a document should be modeled using a log function thus:

$$TF_{l:td} = 1 + \ln(1 + \ln(tf_{td})) \tag{18}$$

Following BM25+, they add δ to ensure there is a sufficient gap between the 0th and 1st term occurrence. They refer to this as $TF_\delta$:

$$TF_{\delta:td} = tf_{td} + \delta \tag{19}$$

For document length normalization they prefer the pivoted component from BM25, equation (7). They refer to this as $TF_p$:

$$TF_{p:td} = \frac{tf_{td}}{1 - b + b . \left(\frac{L_d}{L_{avg}}\right)} \tag{20}$$

Using their rules of combination, a combined $TF_l$, $TF_\delta$, and $TF_p$, gives $TF_{l\circ\delta\circ p}$ defined as:

$$TF_{l\circ\delta\circ p} = 1 + \ln\left(1 + \ln\left(\frac{tf_{td}}{1 - b + b . \left(\frac{L_d}{L_{avg}}\right)} + \delta\right)\right) \tag{21}$$

Finally, adding the IDF component, for which they use:

$$IDF_t = \log\frac{N + 1}{df_t} \tag{22}$$

We assume log is the natural log, giving $TF_{l\circ\delta\circ p}$×ID as:

$$rsv_q = \sum_{t \in q} \ln\frac{N + 1}{df_t} . \left(1 + \ln\left(1 + \ln\left(\frac{tf_{td}}{1 - b + b . \left(\frac{L_d}{L_{avg}}\right)} + \delta\right)\right)\right) \tag{23}$$

Experiments with $TF_{l\circ\delta\circ p}$×ID and other combinations of functions conducted on the Robust04 and WT10g suggest that $TF_{l\circ\delta\circ p}$×ID outperforms BM25.

# 4. LANGUAGE MODELS

Petri *et al.* [17] provide a derivation of Language Models with Dirichlet smoothing that includes document and query length priors – we use this derivation.

Puurula [20] examines improvements in language models, showing three areas where language models have shown improvement. These are the Pitman-Yor Process smoothing, TF×IDF feature weighting, and model-based feedback. The former two are discussed in Sections 4.2 and 4.3, the latter in Section 5.2.

## 4.1 LM-DS

Dirichlet smoothing is a standard method of interpolating between the collection model and the document model. Petri *et al*. [17] give the derivation as:

$$rsv_q = L_q . \log\left(\frac{\mu}{L_d + \mu}\right) + \sum_{t \in q} tf_{tq} . \log\left(\frac{tf_{td}}{\mu} . \frac{L_c}{cf_t} + 1\right) \tag{24}$$

Where $L_q$ is the length of the query, $\mu$ is a tuning parameter, $tf_{tq}$ is the number of times the term occurs in the query, $L_c$ is the length of the collection (in terms), and $cf_t$ is the number of times the term occurs in the collection (the collection frequency). Experiments on .GOV and TREC 7+8 show that this function, LM-DS, outperforming BM25.

## 4.2 LM-PYP

The Pitman-Yor Process (PYP) is used for probabilistic modeling of distributions that follow a power law. Inference on a PYP can be efficiently approximated by combining *power-law discounting* with a Dirichlet-smoothed language model [15, 20]. This discounts term frequencies by:

$$tf'_{td} = \max(tf_{td} - g . {tf_{td}}^{g}, 0) \tag{25}$$

where $g$ is a discounting parameter. This discounting applies to all term frequencies used throughout the ranking function – including the computation of document length, $L'_d$.

Puurula uses this discounting with document length priors[1]:

$$prior_d = \log\left(1 - \frac{L'_d}{L_d + \mu}\right) \tag{26}$$

Where $\mu$ is a smoothing parameter.

For each term in the query the discounted term frequency is used along with the number of times the term occurs in the collection (the collection frequency, $cf_t$,) and the length of the collection measured in terms, $L_c$ to compute the influence of that term with respect to the query, $rsv_{tq}$:

$$rsv_{tq} = tf_{tq} . \log\left(\frac{tf'_{td} . L_c}{\mu . cf_t} + 1\right) \tag{27}$$

Combining these with a query length, $L_q$, gives LM-PYP:

$$rsv_q = L_q . \log\left(1 - \frac{L'_d}{L_d + u}\right) + \sum_{t \in q} tf_{tq} . \log\left(\frac{tf'_{td} . L_c}{\mu . cf_t} + 1\right) \tag{28}$$

Experiments conducted on TREC 1-5, FIRE 2008-2011, and OHSUMED show improvements on LM-DS.

## 4.3 LM-PYP-TFIDF

TF×IDF weighting can be applied to all term frequency values before PYP smoothing. Doing so weights term frequencies based on collection properties [20]. This weighting is done by universally applying the TF×IDF transform to all frequency parameters. For example, the Pitman-Yor TF×IDF smoothed term frequency, $tf''_{td}$, is derived from the TF×IDF–weighted term frequency, $tf'_{td}$ thus:

$$tf''_{td} = \max(tf'_{td} - g . {tf'_{td}}^{g}, 0) \tag{29}$$

where

$$tf'_{td} = \log\left(1 + \frac{tf_{td}}{L_{0d}}\right) . \log\left(\frac{N}{df_t}\right) \tag{30}$$

where $L_{0d}$ is the document $L_0$ norm (the unique term count).

The length of the document is $L''_d$ derived likewise:

$$L''_d = \sum_{t \in d} tf''_{td} \tag{31}$$

and also

$$L'_d = \sum_{t \in d} tf'_{td} \tag{32}$$

Collection smoothing and IDF weighting have overlapping roles and should not both be applied [20]. Using a uniform background distribution for smoothing replaces the collection frequency component of the equation by $N_0$, the number of unique terms in the collection, resulting in a term specific *rsv* component:

$$rsv_{tq} = tf'_{tq} . \log\left(\frac{tf''_{td} . N_0}{\mu} + 1\right) \tag{33}$$

where $tf'_{tq}$ is computed from the query frequency, $tf_{tq}$, and the $L_0$ norm of the query, $L_{0q}$, (the number of unique terms in the query):

$$tf'_{tq} = \log\left(1 + \frac{tf_{tq}}{L_{0q}}\right) . \log\left(\frac{N}{df_t}\right) \tag{34}$$

and $L'_q$ is computed similarly:

$$L'_q = \sum_{t \in d} tf'_{tq} \tag{35}$$

Again, a prior based on document length is used:

$$prior_d = \log\left(1 - \frac{L''_d}{L'_d + \mu}\right) \tag{36}$$

When combined with the query length, $L'_q$, it gives the Pitman-Yor Process TF×IDF ranking function, LM-PYP-TFIDF:

$$rsv_q = L'_q . \log\left(1 - \frac{L''_d}{L'_d + \mu}\right) + \sum_{e \in q} tf'_{tq} . \log\left(\frac{tf''_{td} . N_0}{\mu} + 1\right) \tag{37}$$

When measured on several TREC and FIRE collections this function was shown to outperform LM-DS, and to often outperform the variant without TF×IDF smoothing.

# 5. PSEUDO-RELEVANCE FEEDBACK

The probability ranking principle states that a search engine should order documents from most likely to least likely to be relevant. If successful at this then further processing of the top documents might be used to improve precision and recall. A common approach to this is relevance feedback.

In *full relevance feedback* the search engine presents a results list to the user who selects relevant (and sometimes irrelevant) documents. This input is then used to produce a new list of results. Several approaches exist, some involve performing a new query (for example, *query expansion*) while others simply re-order the results based on the feedback (for example, *query re-ranking*).

[1] These "priors" are not probabilistic priors, they are query term-independent components equal to the log weights of the background smoothing distribution.

If the search engine is effective at ranking then it is reasonable to assume that the top documents are relevant and the bottom documents less so. So the user need not manually identify relevant and irrelevant documents as they are implicit. This approach to feedback is known as *pseudo-relevance feedback*.

Two approaches to pseudo-relevance feedback are examined: query expansion using KL-divergence; and query re-ranking using truncated model-based feedback.

## 5.1 KL-Divergence

The purpose of KL-divergence is to identify how the frequencies in one distribution diverge from the frequencies in a second. When used for relevance feedback, the technique is used to find terms occurring (in the top $k$ documents) more frequently than predicted by collection statistics.

Given a list of results, the top $k$ documents are examined (as if a single meta-document) and the frequency of each unique term is computed. Then, for each term, the Kullback–Leibler divergence from the collection language model is computed:

$$D_{KL}(d||c) = p(d).\log\frac{p(d)}{p(c)} \tag{38}$$

Where $d$ is the meta-document and $c$ is the collection. The probabilities are computed using the maximum likelihood estimation, the observed frequency divided by observed length.

All terms in the meta-document are then ranked from highest to lowest $D_{KL}(d||c)$. The top $m$ terms are used for query expansion. There are many different approaches, but the approach seen in ATIRE is based on Rocchio.

Within the vector space model Rocchio [24] suggests moving the query towards the centroid of known relevant documents and away from the centroid of known non-relevant documents, but keeping the original query, $q_i$. That is,

$$q_{i+1} = \alpha q_i + \beta\frac{\sum_{i=1}^{r} R_i}{r} - \gamma\frac{\sum_{j=1}^{\bar{r}} \bar{R}_j}{\bar{r}} \tag{39}$$

where the new query, $q_{i+1}$, is $\alpha$ proportion of the original query, $\beta$ proportion of the centroid of the $r$ number of relevant documents, $R$, while moving away from the $\bar{r}$ number of non-relevant documents, $\bar{R}$, by $\gamma$ amount.

The ATIRE combination ignores non-relevant documents as they cannot be known ($\gamma = 0$) and it sets $\alpha = \beta = 1$.

Robertson *et al*., [21] assume that documents at the bottom of the results list are non-relevant. However an alternative source of non-relevant documents is the set of documents not in the results lists. We leave for future work the experiment to determine whether using non-relevant documents is effective in feedback.

In summary, the ATIRE relevance feedback approach generates the new query, $q_{i+1}$ from the old query $q_i$ by adding to the old query the top $n$ terms from the top $k$ documents, computed using KL-divergence from the document collection. This can result in the same term occurring in the feedback query multiple times, but that is assumed to be taken care of by the ranking function. As new terms can be added to the query, this is query expansion.

## 5.2 Truncated Model-based Feedback

Puurula [20] suggests re-computing the query frequency of each term as a linear interpolation of the language model of the top $k$ documents and the language model of the document. As this method does not add new terms to the query, it is query re-ranking (albeit by performing a second search). For efficiency reasons, the model uses only the top $k$ documents and only terms present in the query. Both strategies are common with pseudo-relevance feedback for language models.

The language model scores computed by the search engine are log probabilities, so it is necessary to convert out of logs before use,

$$tf'_{tq} = (1-\lambda)\frac{tf_{tq}}{L_q} + \lambda\sum_{d\in R}\frac{e^{rsv_q + rsv_{tq} + prior_d}}{Z} \tag{40}$$

where $R$ is the set of the top $k$ documents, $\lambda$ is a tuning parameter, and $Z$ is introduced as a normalizer,

$$Z = \sum_{d\in R}\sum_{t\in q} e^{rsv_q + rsv_{tq} + prior_d} \tag{41}$$

The normalizer $Z$ is defined so that it can be efficiently computed, by iterating the top documents $d$ from an inverted index for the terms, $t$, in the query. The new query length, $L'_q$ is computed as the sum of $tf'_{tq}$ scores.

This feedback method has been applied to the Pitman-Yor and TF×IDF Pitman-Yor language models of Section 4.2 and 4.3. When tested on the early TREC and FIRE collections, it normally outperformed the same function without feedback.

# 6. EXPERIMENTS

A substantial number of ranking functions are discussed in Sections 3 and 4. Two different feedback method are discussed in Section 5. Although the ATIRE feedback method can be applied to any of the functions, the language model feedback approach relies on the *rsv* scores being log probabilities which BM25 and related functions do not produce.

## 6.1 Training

Each of the ranking functions was trained on the INEX Wikipedia document collection using the title fields of the 68 topics from 2009. This Wikipedia collection is a dump of taken on 8th October 2008 and consists of 2,666,190 articles converted into XML and annotated using the 2008-w40-2 version of YAGO. It is about 50.07GB in size and has been used extensively at INEX.

In all cases, the best parameters were found using particle swarm optimization (64 particles, 20 generations, $\omega$=0.4, $\varphi_p$=0.3, $\varphi_g$=0.2). For the BM25 functions, the search for $b$ was in the range 0 to 1; as was $\delta$; while the search for $k_1$ was from 0 to 3 (all to one decimal place). For LM-DS, the search for $\mu$ was in the range 100 to 3000. For LM-PYP the search for $g$ was in the range 0 to 0.9 in steps of 0.1 (to one decimal place) and $\mu$ from 100 to 3000. For LM-PYP-TFIDF the search for $\mu$ was from 0 to 1 for $g$ it was 0.000 to 0.009 in steps of 0.001 (to one significant figure).

Table 1 presents the best parameters found for each of the ranking functions. Column 1 lists the function, column 2 gives the learned parameters. Column 3 gives the MAP at 1000 documents found during the search – the Oracle MAP on this collection. Column 4 is discussed in Section 6.2. For example, the best parameters for ATIRE BM25 are $b$=0.3 and $k_1$=1.1 which result in a MAP of 0.3502. The highest scoring functions, were BM25+ and BM25L with a MAP of 0.3540 (shown in bold).

It is striking that there is very little variance in the MAP score of the best functions, a result similar to the finding of Petri *et al*. [18] when comparing ATIRE, Indri, and NewSys on .GOV2.

**Table 2: Results (MAP at 1000 scores) from training on INEX 2009 and testing on the 9 TREC collections.**

| Function | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 8 wt2g |
|---|---|---|---|---|---|---|---|---|---|
| ATIRE BM25 | 0.1874 | **0.1795** | 0.2125 | 0.1496 | 0.1288 | 0.1821 | 0.1837 | 0.2112 | 0.2631 |
| BM25L | **0.1875** | 0.1793 | 0.2053 | 0.1238 | 0.1295 | 0.1841 | 0.1874 | 0.2121 | **0.2705** |
| BM25+ | 0.1871 | 0.1776 | 0.2059 | 0.1272 | 0.1307 | 0.1847 | 0.1888 | 0.2146 | 0.2688 |
| BM25-adpt | 0.1835 | 0.1726 | **0.2257** | 0.1511 | **0.1415** | **0.1865** | **0.1951** | **0.2217** | 0.2685 |
| BM25T | 0.1863 | 0.1765 | 0.2024 | 0.1334 | 0.1334 | 0.1857 | 0.1919 | 0.2044 | 0.2663 |
| $TF_{1\cdot\delta\cdot p}$×IDF | 0.1864 | 0.1755 | 0.2070 | 0.1347 | 0.1307 | 0.1834 | 0.1916 | 0.2152 | 0.2671 |
| LM-DS | 0.1766 | 0.1651 | 0.1918 | **0.1620** | 0.1347 | 0.1808 | 0.1874 | 0.2156 | 0.2506 |
| LM-PYP | 0.1698 | 0.1571 | 0.1512 | 0.0606 | 0.0948 | 0.1629 | 0.1853 | 0.2133 | 0.2599 |
| LM-PYP-TFIDF | 0.1737 | 0.1621 | 0.1793 | 0.1030 | 0.1304 | 0.1831 | 0.1946 | 0.2180 | 0.2550 |

The training suggests that if improvements on BM25 are to be seen then those improvements are not going to come from a ranking function. Such improvements might come from recall enhancing tools such as stemming or relevance feedback; or false-positive reducing tools such as stopping; or bigram and proximity processing. Alternatively, the binary assessments used for MAP computation may not be of sufficient granularity to measure the subtle re-orderings seen between these ranking functions; as suggested by Trotman & Keeler [28].

**Table 1: Results of training on the INEX 2009 collection (where MAP at 1000 scores are Oracle scores) and testing on INEX 2010.**

| Function | Parameters | Training INEX 2009 | Testing INEX 2010 |
|---|---|---|---|
| ATIRE BM25 | b=0.3; $k_1$=1.1 | 0.3502 | 0.3460 |
| BM25L | b=0.3; $k_1$=1.8; δ=0.6 | **0.3540** | 0.3501 |
| BM25+ | b=0.3; $k_1$=1.6; δ=0.7 | **0.3540** | 0.3556 |
| BM25-adpt | b=0.3 | 0.3475 | 0.3429 |
| BM25T | b=0.3 | 0.3452 | **0.3560** |
| $TF_{1\cdot\delta\cdot p}$×IDF | b=0.3; δ=0.5 | 0.3536 | 0.3547 |
| LM-DS | μ=1089 | 0.3489 | 0.3395 |
| LM-PYP | μ=1303; g=0.2 | 0.3440 | 0.3432 |
| LM-PYP-TFIDF | μ=0.53; g=0.004 | 0.3406 | 0.3341 |

## 6.2 Same Documents Different Queries

To measure the performance on untrained queries the ranking functions and parameters from Table 1 were tested on the INEX 2010 topics against the same document collection. The title field of the 52 topics was used. The results are presented in the final column of Table 1. For example ATIRE BM25 with $k_1$=1.1 and $b$=0.3 scores a MAP of 0.3460 when trained on INEX 2009 and tested on INEX 2010. The highest scoring function was BM25T with a MAP of 0.3560 (in bold). It is again striking how little variance there is across the different ranking functions.

## 6.3 Different Documents Different Queries

To measure the portability of the ranking functions, the ranking functions and parameters from Table 1 were tested on each of the first 8 TREC *ad hoc* collections and the TREC 8 wt2g collection.

Table 3 lists the topics and the document collections that were used. The first row lists the TREC campaign, the second gives the topic numbers, and the third provides the source of the documents. For example, TREC 7 used topics 351-400 on the documents shipped on TREC disks 4 and 5, but without the *Congressional Record* documents. The title field was used as the query (TREC 4 does not have titles so the description field was used). The MAP at 1000 scores are presented in Table 2**.** The first column gives the name of the ranking function, the second and subsequent columns give the MAP at 1000 score seen for the given TREC campaign. For example, ATIRE BM25 scores a MAP at 1000 of 0.1874 on TREC 1. The best scores for each TREC are shown in bold.

When tested in this way BM25-adpt outperformed the others on 5 of the 9 collections. The BM25 functions generally performed better than language models – but ATIRE was designed as a BM25 based search engine, and the language models were not implemented by the original authors.

BM25-adpt, which computed a term-specific $k_1$ from the index, does appear to be easily transferable from collection to collection without retraining.

**Table 3: TREC collections used in the experiments. †TREC 7 and 8 used disks 4 and 5, without the Congressional Record.**

| TREC | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 8 wt2g |
|---|---|---|---|---|---|---|---|---|---|
| Topics | 51-100 | 101-150 | 151-200 | 201-250 | 251-300 | 301-350 | 351-400 | 401-450 | 401-450 |
| Disks | 1+2 | 1+2 | 1+2 | 2+3 | 2+4 | 4+5 | 4+5† | 4+5† | wt2g |

## 6.4 Feedback

The ranking functions from Table 1, with relevance feedback added, were re-retained using a 50-generation particle swarm optimizer. Not only were $n$ (terms) and $k$ (documents) learned for the KL-Divergence feedback algorithm, but two sets of ranking function parameters were learned, the first was the pre-feedback search parameters while the second was the post-feedback search parameters. INEX 2009 was used for training and INEX 2010 for testing. Both $k$ and $n$ were searched for in the range 1 to 100.

The results are presented at the top of Table 4. Column 1 lists the function name, column 2 lists the MAP at 1000, and column 3 lists the score when those parameters are used with the 2010 topics. For brevity the optimal parameters (of which there are up-to 8) are omitted. For example, the best score, shown in bold, during training was for BM25+ which scores 0.3887 during training and 0.3728 during testing.

The bottom of Table 4 presents, for the language models ranking algorithms, the scores seen when used with the truncated model-based feedback method. The search for $k$ was in the range 1 to 100 while $\lambda$ in the range 0 to 1.

The results show improvements on BM25 based ranking functions when query expansion is used, but not (italicized) in the language model functions. Improvements are sometimes seen in language models when truncated model-based feedback is used.

**Table 4: Feedback trained on INEX 2009 and tested on INEX 2010. Top shows KL-divergence. Bottom shows truncated model-based feedback. †Significantly better ($p$=0.0267) than best in Table 1**

| Function | Training INEX 2009 | Testing INEX 2010 |
|---|---|---|
| ATIRE BM25 | 0.3692 | 0.3646 |
| BM25L | 0.3842 | 0.3686 |
| BM25+ | **0.3887** | 0.3728 |
| BM25-adpt | 0.3766 | 0.3679 |
| BM25T | 0.3761 | 0.3638 |
| $TF_{1+\delta+p}$×IDF | 0.3730 | **0.3762**† |
| LM-DS | *0.3411* | *0.3320* |
| LM-PYP | *0.2668* | *0.2199* |
| LM-PYP-TFIDF | *0.3288* | *0.3186* |
| LM-DS | *0.3468* | 0.3431 |
| LM-PYP | **0.3519** | **0.3534** |
| LM-PYP-TFIDF | *0.3385* | *0.3267* |

## 6.5 Stemming and Stopping

Many different stemming algorithms are seen in the literature, including the simple s-stripper [5], and those of Porter [19], Paice [16], Lovins [9], and Krovetz [7]. Many different stop word lists are also seen in the literature including: the 313 word list of the NCBI; and the 988 word list used by Puurula [20].

Re-training each ranking function against each of these stemmers and stop word lists with each of the feedback mechanism is prohibitive. It is, however, possible to test one ranking function. BM25-adpt, BM25T, and LM-DS each take a single parameter, but BM25-adpt outperformed the others on the TREC collections, and so it was chosen. Both stemming and stopping were performed before indexing and so document lengths do not include stopped works and term frequencies include all word forms.

Table 5 presents the results of training BM25-adpt with stopping and stemming on the INEX 2009 collection. The first row gives the name of the stemmer, the first column gives the number of words in the stop word list. The other cells give the MAP at 1000. For example, with the s-stripper and without stopping BM25-adpt achieved a score of 0.3549, the best score seen (shown in bold).

The table shows two patterns on this collection with this ranking function: first that the smaller the stop word list the better the performance (with no stopping being best); and second that the s-stripper is more effective than the others, suggesting that weak stemming is better than strong stemming. Only the Krovetz stemmer and the s-stripper were better than not stemming. Others [20] suggest that stop words are important for language models and for natural language processing, but we only tested BM25-adpt.

**Table 5: Training of BM25-adpt on INEX 2009 with different stemmers and stop word lists. Scores are MAP at 1000.**

| Stopwords | None | Krovetz | Lovins | Porter | Paice | S |
|---|---|---|---|---|---|---|
| **0** | 0.3475 | 0.3528 | 0.2417 | 0.3388 | 0.3109 | **0.3549** |
| **313** | 0.3170 | 0.3485 | 0.2321 | 0.3345 | 0.3060 | 0.3514 |
| **988** | 0.2999 | 0.3357 | 0.2172 | 0.3150 | 0.2907 | 0.3230 |

## 6.6 Stemming and Feedback

Section 6.4 suggests that feedback is effective and Section 6.5 suggest that stemming is effective (and that stopping is not). In this section the combination of the two is examined. That is, the ranking functions are re-trained on an index stemmed with the s-stripper but no stop word removal. The INEX 2009 collection was again used for training and INEX 2010 for testing.

Table 6 presents the results. Column 1 lists the function, column 2 the MAP at 1000 seen in training and column 3 likewise for testing. The table shows that, when training, stemming with feedback is almost always better than feedback alone. The two cases where it is not (BM25-adpt, and LM-PYP-TFIDF) are show in italics. In almost all cases, the scores on the test queries are better than those seen without stemming.

**Table 6: Feedback with stemming trained on INEX 2009 and tested on INEX 2010. Top shows KL-divergence. Bottom shows truncated model-based feedback. †Significantly better ($p$=0.0292) than best in Table 4 and ($p$<0.0001) Table 1**

| Function | Training INEX 2009 | Testing INEX 2010 |
|---|---|---|
| ATIRE BM25 | 0.3815 | 0.3840 |
| BM25L | 0.3925 | 0.4016 |
| BM25+ | **0.3971** | 0.3977 |
| BM25-adpt | *0.3698* | 0.3840 |
| BM25T | 0.3955 | **0.4054**† |
| $TF_{1+\delta+p}$×IDF | 0.3845 | 0.4011 |
| LM-DS | 0.3520 | 0.3423 |
| LM-PYP | 0.2780 | 0.2493 |
| LM-PYP-TFIDF | *0.3039* | *0.2607* |
| LM-DS | 0.3631 | 0.3698 |
| LM-PYP | **0.3638** | **0.3733** |
| LM-PYP-TFIDF | 0.3524 | 0.3621 |

## 6.7 Stemming, Feedback, TREC

In the final experiment the parameters learned in section 6.6 where tested on the TREC collections in Table 3 with the s-stemmer and without removal of stop words. The results are presented in Table 7. The first column lists the ranking function and the first row the TREC collection. The remaining cells give the MAP at 1000 for that function against that collection. The best scores for each collection are given in bold. The cases where stemming with feedback performed worse than the function on its own (Table 3 cf. Table 7) are shown in italics.

The general trends seen before continue. Stemming appears to be effective; query expansion is not effective with language models, but is with BM25; truncated model-based feedback is effective with language models. There is no clear best function.

## 6.8 Statistical Significance

Indiscriminately performing hundreds of significance tests is unlikely to show any meaningful result. However a paired 1-tailed $t$-test was used to compare the best in Tables 1, 4, and 6. Feedback is better than no feedback ($p$=0.0267). Stemming with feedback is better than just feedback ($p$=0.0292). Stemming with feedback is better than neither ($p$<0.0001). No adjustments (e.g. Bonferroni) were made.

# 7. CONCLUSIONS

This investigation examined 9 ranking functions, 2 relevance feedback methods, 5 stemming algorithms, and 2 stop word lists. It shows that stop words are ineffective, that stemming is effective, that relevance feedback is effective, and that the combination of not stopping, stemming, and feedback typically leads to improvements on a plain ranking function. However, there is no

**Table 7: Results (MAP at 1000 scores) from training on INEX 2009 with feedback and stemming; tested on TREC collections.**

| Function | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 8 wt2g |
|---|---|---|---|---|---|---|---|---|---|
| ATIRE BM25 | 0.2221 | **0.2103** | 0.2581 | 0.1779 | 0.1585 | 0.2348 | 0.2021 | 0.2342 | 0.2782 |
| BM25L | 0.2236 | 0.2081 | 0.2485 | 0.1446 | 0.1627 | **0.2472** | 0.2157 | 0.2420 | 0.2938 |
| BM25+ | 0.2215 | 0.2085 | 0.2577 | **0.1858** | 0.1746 | 0.2464 | **0.2236** | **0.2466** | 0.2929 |
| BM25-adpt | 0.2141 | 0.1923 | **0.2610** | 0.1718 | **0.1748** | 0.2417 | 0.2085 | 0.2441 | 0.2907 |
| BM25T | 0.2160 | 0.2057 | 0.2525 | 0.1628 | 0.1559 | 0.2257 | *0.1838* | 0.2359 | **0.2976** |
| $TF_{I\text{-}\delta\text{-}p}$×IDF | **0.2242** | 0.2091 | 0.2530 | 0.1564 | 0.1625 | 0.2424 | 0.2136 | 0.2421 | 0.2869 |
| LM-DS | 0.1954 | 0.1839 | 0.2144 | *0.1551* | 0.1467 | 0.2281 | 0.1943 | 0.2236 | *0.2281* |
| LM-PYP | *0.1084* | *0.0605* | *0.0543* | *0.0280* | *0.0193* | *0.1137* | *0.1461* | *0.1541* | *0.1300* |
| LM-PYP-TFIDF | *0.1596* | *0.1612* | *0.1690* | *0.0945* | *0.1077* | *0.1741* | 0.2173 | *0.1968* | 0.2592 |
| LM-DS | 0.2006 | 0.1858 | 0.2249 | *0.1570* | 0.1639 | 0.2267 | 0.1936 | 0.2222 | 0.2542 |
| LM-PYP | 0.1803 | *0.1514* | 0.1584 | 0.0666 | 0.1043 | 0.2087 | 0.1919 | 0.2236 | 0.2604 |
| LM-PYP-TFIDF | 0.1818 | 0.1639 | 0.1800 | *0.0588* | 0.1422 | 0.2293 | 0.1994 | 0.2282 | 0.2640 |

clear evidence that any one of the ranking functions is systematically better than the others.

The implementations were all done by one author and into a BM25-based search engine, which may negatively affect the performance of language modelling. Training proved difficult with so many parameters and it is likely that the best parameters were not found. Specifically, the sensitivity of the power law discounting parameter, *g*, in PYP and PYP-TFIDF was an issue.

Comparison of large numbers of ranking functions is exploratory in nature due to the number of observed effects, but we found no one ranking function consistently outperforming the others.

The use of these ranking functions for scenarios such as learning-to-rank was not explored – we leave that investigating for future work. We do, however, observe that in our scenario BM25-based functions appear to generally outperform language modelling.

# REFERENCES


[1] Armstrong, T.G., A. Moffat, W. Webber, J. Zobel, Has adhoc retrieval improved since 1994? *SIGIR 2009*, p. 692-693.

[2] Armstrong, T.G., A. Moffat, W. Webber, J. Zobel, Improvements that don't add up: ad-hoc retrieval results since 1998, *CIKM 2009*, p. 601-610.

[3] Gövert, N., G. Kazai. Overview of the INitiative for the Evaluation of XML retrieval (INEX) 2002, *INEX 2002,* p. 1-17

[4] Harman, D. Overview of the First TREC Conference. *SIGIR 1993*, p. 36-47

[5] Jia, X.-F., D. Alexander, V. Wood, A. Trotman, University of Otago at INEX 2010, *INEX 2010*, p. 250-268.

[6] Kando, N., K. Kuriyama, T. Nozue, K. Eguchi, H. Kato, S. Hidaka, Overview of IR Tasks at the First NTCIR Workshop, *NTCIR-1*, (1999), p. 11-44.

[7] Krovetz, R. Viewing morphology as an inference process, *SIGIR 1993*. p. 191-202

[8] Lee, L., IDF revisited: a simple new derivation within the Robertson-Sparck Jones probabilistic model, SIGIR 2007, p. 751-752.

[9] Lovins, J.B., Error evaluation for stemming algorithms as clustering algorithms. *JASIS*, (1971), 22(1):28-40.

[10] Lv, Y., C. Zhai, Adaptive term frequency normalization for BM25, *CIKM 2011*, p. 1985-1988.

[11] Lv, Y., C. Zhai, Lower-bounding term frequency normalization, *CIKM 2011*, p. 7-16.

[12] Lv, Y., C. Zhai, When documents are very long, BM25 fails! *SIGIR 2011*, p. 1103-1104.

[13] Lv, Y., C. Zhai, A log-logistic model-based interpretation of TF normalization of BM25, *ECIR 2012*, p. 244-255.

[14] Majumder, P., M. Mitra, D. Pal, A. Bandyopadhyay, S. Maiti, S. Pal, D. Modak, S. Sanyal, The FIRE 2008 Evaluation Exercise. *TALIP*, (2010), **9**(3):1-24.

[15] Momtazi, S., D. Klakow, Hierarchical pitman-yor language model for information retrieval, *SIGIR 2010*, p. 793-794.

[16] Paice, C.D., Another stemmer. *SIGIR Forum*, (1990), 24(3):56-61.

[17] Petri, M., J.S. Culpepper, A. Moffat, Exploring the magic of WAND, *ADCS 2013*, p. 58-65.

[18] Petri, M., A. Moffat, J.S. Culpepper, Score-safe term-dependency processing with hybrid indexes, *SIGIR 2014*, p. 899-902.

[19] Porter, M., An Algorithm for suffix stripping. *Program*, 1980. 14(3):130-137.

[20] Puurula, A., Cumulative Progress in Language Models for Information Retrieval, *ALTA 2013*, p. 96-100.

[21] Robertson, S.E., S. Walker, M. Beaulieu, Okapi at TREC-7: automatic ad hoc, filtering, VLC and interactive, *TREC-7,* (1998), p. 253-264.

[22] Robertson, S.E., S. Walker, M.M. Beaulieu, M. Gatford, A. Payne. Okapi at TREC-4. *TREC-4* (1995), p. 73-96

[23] Robertson, S.E., S. Walker, S. Jones, M.M. Beaulieu, M. Gatford. Okapi at TREC-3. *TREC-3*, (1994), p. 109-126

[24] Rocchio, J., Relevance feedback in information retrieval. In *The Smart Retrieval System- Experiments in Automatic Document Processing*, G. Salton (Ed). (1971), p. 313-323.

[25] Rousseau, F., M. Vazirgiannis, Composition of TF normalizations: new insights on scoring functions for ad hoc IR, *SIGIR 2013*, p. 917-920.

[26] Singhal, A., C. Buckley, M. Mitra. Pivoted document length normalization. *SIGIR 1996*, p. 21-29

[27] Trotman, A., X. Jia, M. Crane, Towards an Efficient and Effective Search Engine, *SIGIR 2012 Workshop on Open Source Information Retrieval*, p. 40-47.

[28] Trotman, A., D. Keeler, Ad hoc IR: not much room for improvement, *SIGIR 2011*, p. 1095-1096.