About Learning Models with Multiple Query Dependent Features

CRAIG MACDONALD, RODRYGO L.T. SANTOS and IADH OUNIS
University of Glasgow, UK
BEN HE
University of Chinese Academy of Sciences

Several questions remain unanswered by the existing literature concerning the deployment of query dependent features within learning to rank. In this work, we investigate three research questions to empirically ascertain best practices for learning to rank deployments: (i) Previous work in data fusion that pre-dates learning to rank showed that while different retrieval systems could be effectively combined, the combination of multiple models within the same system was not as effective. In contrast, the existing learning to rank datasets (e.g. LETOR), often deploy multiple weighting models as query dependent features within a single system, raising the question as to whether such combination is needed. (ii) Next, we investigate whether the training of weighting model parameters, traditionally required for effective retrieval, is necessary within a learning to rank context. (iii) Finally, we note that existing learning to rank datasets use weighting model features calculated on different fields (e.g. title, content or anchor text), even though such weighting models have been criticised in the literature. Experiments to address these three questions are conducted on Web search datasets, using various weighting models as query dependent, and typical query independent features, which are combined using three learning to rank techniques. In particular, we show and explain why multiple weighting models should be deployed as features. Moreover, we unexpectedly find that training the weighting model's parameters degrades learned models effectiveness. Finally, we show that computing a weighting model separately for each field is less effective than more theoretically-sound field-based weighting models.

Received 11th April 2012; revised 17th September 2012; revised 17 December 2012; accepted 6th February 2013.

Categories and Subject Descriptors: H3.3 [**Information Storage & Retrieval**]: Information Search & Retrieval

General Terms: Performance, Experimentation

Additional Key Words and Phrases: Learning to rank, Samples, Field-based weighting models

1. INTRODUCTION

Numerous statistical document weighting models have been proposed for information retrieval (IR) systems, dating back to TF.IDF [Robertson and Jones 1976] and beyond, which are able to weight the occurrence of query terms within a document,

Author's address: C. Macdonald & I. Ounis, School of Computing Science, University of Glasgow, Lilybank Gardens, G12 8QQ, Glasgow, Scotland, UK

such that an effective ranking of documents can be obtained. Chief among the predominant families include Okapi BM25 [Robertson et al. 1992], language modelling [Zhai and Lafferty 2001] and Divergence From Randomness (DFR) [Amati 2003], which have all demonstrated effectiveness across a variety of test collections. Different weighting models have different properties, by modelling the number of occurrences of a term in a document of a given length - in contrast to the occurrences of the term in the entire corpus - in different manners. For instance, Fang et al. [2004] identified different heuristics that the most effective weighting models encapsulate to different extents.

The ability of different weighting models to identify both the same and different relevant documents was first exploited by data fusion/metasearch approaches. For example, Vogt and Cottrell [1998] noted that the chorus effect of agreement between different IR systems should increase the expected relevance of a document. Similarly, some attempts were made to use data fusion to combine retrieval scores from different document representations (also known as fields, e.g. title, content and the anchor text from incoming hyperlinks) within a single retrieval system [Ogilvie and Callan 2003; Kamps 2006]. However, Robertson et al. [2004] warned against the linear combination of document weighting model scores for different representations, due to the non-linear nature of term frequency saturation in weighting models. Indeed, *field-based* models have since been proposed as a sound and more effective alternative (e.g. BM25F [Zaragoza et al. 2004] or PL2F [Macdonald et al. 2006]).

Learning to rank is closely related to data fusion, whereby different features computed for a sample of documents are appropriately combined within an effective *learned model*. In this work, we study how conclusions from earlier literature including data fusion studies can be transferred to learning to rank. Indeed, we note that the current deployments of learning to rank seem to be in contrast to earlier literature, and use these contrasts to posit several questions. For instance, despite concerns raised in the literature, many learning to rank settings have combined multiple different weighting models within a single learned model, or even the combination of the same weighting model for different document representations [Liu 2009]. Moreover, the various weighting models contain parameters (e.g. controlling document length normalisation) that can significantly impact on their retrieval performance [He and Ounis 2003; He et al. 2008]. Historically, these parameters have been *trained*[1] to determine values that result in an effective performance [Chowdhury et al. 2002]. Yet, it is unclear if training these parameters has a significant impact when weighting models are combined within a learned model. Similarly, within a learning to rank setting, it is important to determine if a field-based model is more effective than the combination of models representing different fields.

Hence, in this work, we experiment to examine:

—the role of multiple weighting models within an effective learned model.

—the role of the training of the parameters of weighting models within an effective learned model.

—the role of field-based models within an effective learned model.

[1]In this work, we differentiate between the training of parameters of features, and the learning of combinations of features.

We not only experimentally examine these roles, but analyse to determine the reason behind our observations. Thorough experiments are conducted on two datasets, namely the TREC 2009-2011 Web track and the corresponding test collection of 50 million Web documents, and the LETOR v4.0 learning to rank dataset, which is based on the TREC 2007 and 2008 Million Query track and corresponding test collection of 25 million Web documents. To enable thorough experimentation, we employ an IR system adapted for the efficient calculation of multiple query dependent features, which we call the *fat* framework. The fat framework is so-called because it "fattens" the result set from the initial ranking (known as the *sample* [Liu 2009]) with the postings of matching terms from the inverted index. By doing so, it allows additional features to be later computed for the application of a learned model, without resorting to subsequent access(es) of the inverted index. Hence, the framework provides advantages for the experiments in this paper, such as the easy calculation of additional query dependent features for an existing document sample, and the easy training of query dependent features.

From the observations arising from our experiments, we make recommendations for the practical, effective deployment of weighting models within a learning to rank setting, suitable for both learning to rank deployments and future datasets. For instance, our results attest the effectiveness of using multiple weighting models within a learned model. Moreover, we show how different features can be characterised as chorus (a reinforcing feature), skimming (a contrasting feature) or dark horse (an occasionally useful feature), as originally proposed by Vogt and Cottrell [1998]. Next, contrary to previous evidence in the literature, we do not find any significant evidence that there is a need to train the length normalisation parameters of weighting models to attain an effective learned model, and explain why the learning to rank techniques are able to account within their learning processes for any document length biases present in the features. Furthermore, we find that current learning to rank datasets do not deploy the most effective approach for creating features from document fields.

In light of the above research questions and findings, the central contributions of this paper are that we revisit the historically accepted best practices for weighting models, and investigate their impact in a modern learning to rank setting. The remainder of this paper is structured as follows: Section 2 reviews literature on learning to rank. Section 3 introduces the problem of combining multiple weighting models that this work addresses, along with the tackled research questions. This is followed in Section 4 by the details of our experimental setup. Sections 5 - 7 contain the results and analysis for each of our three research questions; Finally, Section 8 provides concluding remarks.

2. BACKGROUND

Recently, the idea of learning a ranking model has become a predominant research interest and a deployed approach for effective information retrieval systems [Liu 2009]. In particular, many learning to rank approaches attempt to learn a combination of feature values (called the *learned model*). The resulting learned model is applied to a vector of feature values for each document, to determine the final scores for producing the final ranking of documents for a query.

Regardless of the applied technique, the general steps for obtaining a learned model using a learning to rank technique are the following [Liu 2009]:

1. Top k Retrieval: For a set of training queries, generate a *sample* of k documents using an initial retrieval approach.
2. Feature Extraction: For each document in the sample, extract a vector of feature values. A feature is a binary or numerical indicator representing the quality of a document, or its relation to the query.
3. Learning: Learn a model by applying a learning to rank technique. Each technique deploys a different loss function to estimate the goodness of various combinations of features. Documents are labelled according to available relevance assessments.

Once a learned model has been obtained from the above learning steps, it can be deployed within a search engine as follows:

4. Top k Retrieval: For an unseen test query, a sample of k documents is generated in the same manner as in step (1),
5. Feature Extraction: As in step (2), a vector of feature values is extracted for each document in the sample. The set of features should be exactly the same as for step (2).
6. Learned Model Application: The final ranking of documents for the query is obtained by applying the learned model on every document in the sample, and sorting by descending predicted score.

It is expensive to calculate all features on all documents in the collection, or even all documents matching any query term at retrieval time [Liu 2009]. For this reason, Liu [2009] introduces the notion of the *sample* of top-ranked documents for which all features should be calculated (steps 1 and 4). The sample documents are then re-ranked by the learned model. Typically, a standard weighting model, such as BM25 [Robertson et al. 1992], is used to rank enough documents [Cambazoglu et al. 2010; Liu 2009] to obtain sufficient recall. In this way, the sample is not an unbiased statistical sample of a population, but instead contains a portion of the corpus that is most likely to contain relevant documents for the query.

Once the sample has been identified, the additional features can be extracted or computed (steps 2 or 5). Features can be query dependent (QD), e.g. document weighting models such as BM25 [Robertson et al. 1992] or language modelling [Zhai and Lafferty 2001], or query independent (QI), e.g. URL length [Kraaij et al. 2002] or PageRank [Page et al. 1998].

When the features have been extracted, the model can be learned or applied (steps 3 or 6). For some learners, such as Automatic Feature Selection (AFS) [Metzler 2007b], Adarank [Xu and Li 2007], RankSVM [Joachims 2002] and ListNet[2] [Cao et al. 2007], the learned model takes the form of a linear combination of feature values:

$$score(d, Q) = \sum_{f} \alpha_i \cdot f_{i,d} \tag{1}$$

[2]when a linear Neural Network is applied, as is commonly performed.

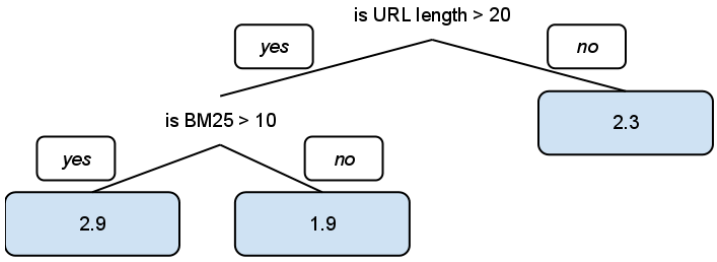

Fig. 1. Example regression tree for a Web search task.

where $f_{i,d}$ is the value of the ith feature for document d, and α_i is the weight of that feature.

On the other hand, the learned model obtained from some other learners takes the form of regression trees [Ganjisaffar et al. 2011; Tyree et al. 2011; Weinberger et al. 2010; Wu et al. 2008]. In particular, the learned model is the linear combination of the outputs from a set of regression trees T, known as an *ensemble*:

$$score(d, Q) = \sum_{t \in T} t(\mathbf{f}_d). \tag{2}$$

Each regression tree t returns a value depending on a series of decisions based on the vector of feature values $\mathbf{f}_d$ for document d. Figure 1 shows an example regression tree for a Web search task. The output value for a particular tree is obtained by traversing the nodes of the tree according to the decisions at each node ("is a feature value higher or lower than a threshold?"), then returning the value at the identified leaf node. In our experiments, we deploy three learning to rank techniques that cover both linear and regression tree types of learned models. These are AFS and RankSVM (both linear), as well as the state-of-the-art LambdaMART technique [Wu et al. 2008], which won the 2011 Yahoo! learning to rank challenge [Chappelle and Chang 2011].

In this paper, we focus on the role of query dependent features for attaining effective learned models. Indeed, while research on query independent features is wide and varied (e.g. [Bendersky et al. 2011; Page et al. 1998]), the role of query dependent features within a learning to rank setting has seen considerably less research. Various types of query dependent feature may exist: traditional document weighting models that act on query terms; proximity models (e.g. [Peng et al. 2007]) that act on pairs of query terms, or even link analysis approaches that consider the linkage patterns within the top-ranked documents (e.g. SALSA [Lempel and Moran 2001]). Our work investigates on the role played by multiple document weighting models – e.g. Okapi BM25, language modelling – and how they bring different evidence within the learning to rank process. Document weighting models are possibly the most studied aspect of information retrieval, and perhaps due to the plethora of proposed models in the literature, we observe that learning to rank datasets often have multiple of such features. We aim to investigate if using multiple weighting models within a learned model enhances effectiveness, or just adds redundancy.

Despite our focus on these query dependent features, we ensure that our experiments are realistic by controlling for the presence of query independent features within the learned models. In Section 3, we study the combination of document weighting models from a historical perspective (e.g. in data fusion), which we contrast with modern learning to rank deployments. Based on the contrasts that we observe, we posit several research questions that permit best practices in learning to rank to be experimentally derived. This is followed in Section 4 by details of the experimental setup that we deploy in this article.

3. COMBINING WEIGHTING MODELS

In the following, we motivate three research questions behind the use of multiple weighting models in learning to rank, namely the combination of multiple weighting models (Section 3.1), whether to train the weighting model parameters (Section 3.2) and how fields should be encompassed by the weighting models deployed in learning to rank (Section 3.3). For each, we identify historical perspectives in the literature that pre-date learning to rank, and contrast these with current practices in learning to rank, to formulate the research question that we later address through experiments.

3.1 Multiple Weighting Models

The combination of weighting models or entire IR systems to produce improved rankings has a long history in IR, stretching back to the initial data fusion work in the first TREC conference [Fox et al. 1993]. Various data fusion/metasearch techniques were proposed, such as CombSUM (a linear combination of the scores of a document from each retrieval system), or CombMNZ (defined as CombSUM multiplied by the number of systems that retrieved the document) [Fox and Shaw 1994], to name but a few. These are effective as they are able to accumulate retrieval evidence from *different* retrieval systems. Indeed, Vogt and Cottrell [1998] enumerated three beneficial effects of the combination of multiple systems:

—**The skimming effect:** different retrieval approaches may retrieve different relevant items.

—**The chorus effect:** several retrieval approaches suggesting the same relevant items provide stronger evidence of the relevance of those items.

—**The dark horse effect:** a retrieval approach may be particularly effective compared to other approaches for retrieving at least some relevant items.

Hence, the diversity of retrieval systems being combined is indicative of the resulting effectiveness [Vogt 1997]. Indeed, Vogt and Cottrell [1998] suggested it was best to linearly combine two IR systems *"when they do not rank relevant documents in a similar fashion"*. On the other hand, the chorus effect assures us that documents retrieved by multiple retrieval approaches are more likely to be relevant - a property which is instilled in the CombSUM and CombMNZ data fusion techniques [Lee 1997]. Indeed, Lee [1997] proposed overlap-based measures to compare the documents retrieved by different systems, and asserted that *"different [systems] retrieve similar sets of relevant documents but retrieve different sets of irrelevant documents"* - a variation of the chorus effect. However, he did not show that such systems were the best to combine.

Given the effectiveness of combining multiple retrieval systems, it makes sense to combine multiple retrieval *approaches* within the same system. Indeed, in the LETOR v3.0 learning to rank dataset (which is based on a TREC Web track test collection) [Qin et al. 2009], many query dependent (QD) weighting model features are computed, such as: TF.IDF, BM25, Jelinek Mercer language modelling and Dirichlet language modelling (see feature IDs 15, 25, 30 and 35 in Table I). Similar to the LETOR datasets, the MSLR-WEB10K and MSLR-WEB30K datasets[3] previously used by the Bing search engine and publicly released by Microsoft Research also deploy multiple document weighting model features. Of the various heuristics identified by Fang et al. [2004] that characterise the effective weighting models, the weighting models in Table I encapsulate most of them to various degrees. Despite these, we note no published evidence examining the role of multiple weighting as features within learning to rank settings. For instance, despite his comprehensive review and experimental comparison of learning to rank techniques, Liu [2009] does not actually show that learning to rank beats the baseline formed by the sample, yet alone the combination of multiple query dependent features. The same applies to the review book of Li [2011]. Instead, it appears to be simply generally accepted that learning to rank techniques should combine multiple query dependent features for effective retrieval, without backing evidence addressing the issue in details.

On the other hand, Beitzel et al. [2004] showed that different retrieval approaches within the same system were considerably more correlated than different systems. Indeed, as the necessary diversity was not attained in combining different weighting models using data fusion techniques, it did not result in improved effectiveness. Similarly, within a learning to rank setting, each of the weighting models is computed using the same retrieval system, using the same document representations and query formulations. Therefore, the weighting models are likely not to present the diversity in results required for effective combination, as suggested by Beitzel et al. [2004]. This highlights a concern in deploying multiple weighting models as features within learning to rank, which we address in this paper.

We also note that there are contrasts between previous studies in the data fusion literature and modern learning to rank deployments. For instance, the data fusion literature draws conclusions about linear combinations of weighting models using data fusion techniques such as CombSUM and CombMNZ. However, in data fusion, the sets of retrieved documents by each system are different, and the number of systems retrieving a given document accounts for at least part of the benefit of data fusion [Lee 1997] (as exemplified by the CombMNZ technique, where the total score of a document across different retrieval approaches is multiplied by the number of systems retrieving a given document). In contrast, in learning to rank, the sample defines the set of documents that will have scores computed for each weighting model feature. Hence, as every sample document is ranked and retrieved, the importance of overlap disappears. Moreover, while the earlier data fusion work has focused on linear combinations of weighting models (c.f. CombSUM), the setting posed by learning to rank can differ. Indeed, as highlighted in Section 2, non-linear learned models, such as regression trees, are often applied.

[3] `http://research.microsoft.com/en-us/projects/mslr/`

ID	Feature Description
...	...
11	TF*IDF of body
12	TF*IDF of anchor
13	TF*IDF of title
14	TF*IDF of URL
15	TF*IDF of whole document
...	...
21	BM25 of body
22	BM25 of anchor
23	BM25 of title
24	BM25 of URL
25	BM25 of whole document
26	LMIR.ABS of body
27	LMIR.ABS of anchor
28	LMIR.ABS of title
29	LMIR.ABS of URL
30	LMIR.ABS of whole document
31	LMIR.DIR of body
32	LMIR.DIR of anchor
33	LMIR.DIR of title
34	LMIR.DIR of URL
35	LMIR.DIR of whole document
...	...

Table I. Selection of weighting model features for the LETOR v3.0 dataset, reproduced from [Qin et al. 2009].

Hence, a question arises concerning the motivation for deploying multiple weighting models within a learned model: If more than one document weighting model can be easily calculated, what is the resulting benefit in effectiveness when deploying many weighting models with different properties compared to just the single model that generated the sample? Indeed, the various publicly released learning to rank datasets all deploy multiple document weighting models, without, to the best of our knowledge, published theoretical or empirical justifications confirming the benefits of doing so. For these reasons, we argue that there is an open question with respect to the combination of weighting models, concerning whether conclusions from data fusion generalise into the more modern learning to rank settings. In particular, does the effectiveness attained by the data fusion of multiple retrieval systems translate into effective learned models when using multiple weighting models as features, or are such weighting models from a single system insufficiently diverse for effective combination, as highlighted by Beitzel et al. [2004]? This provides the basis for the first research question that we tackle in this paper:

RQ 1. *Should multiple document weighting models be deployed as features within a learned model for effective retrieval?*

3.2 Weighting Model Parameters

If adding more than one document weighting model into the feature set is effective, then is there any advantage in training their document length normalisation or smoothing parameters within the features deployed for learning? Much literature has investigated the proper setting of parameters: for example, the classical

method of Singhal et al. [1996] pivots the document length normalisation factor to fit the relevance judgement information. The default settings for BM25 of $k_1 = 1.2$, $k_3 = 1000$ and $b = 0.75$ were obtained by taking into account the relevance judgements in experiments on a merged dataset of TREC collections [Jones et al. 2000]. Indeed, in the past, the setting of such parameters of weighting models has typically been addressed through the *training* (also known as *optimisation*), where the parameter value(s) are automatically found that maximise an evaluation measure on a training dataset. For instance, in their study to ascertain how many relevance assessments were necessary for effective training, He et al. [2008] used simulated annealing [Kirkpatrick et al. 1983] to find length normalisation parameter values. Metzler [2007a] used a more advanced function requiring gradients to find language model smoothing parameters.

Chowdhury et al. [2002] argued that the document length normalisation parameter required recalibration for different collections, and showed that without training, performance could be negatively impacted. Indeed, the stability of document length normalisation or smoothing parameters are key concerns [He and Ounis 2005; Zhai and Lafferty 2001]. Other methods have been proposed to aid in the selection of suitable length normalisation parameter values without excessive training [He and Ounis 2003].

Given the importance of the proper training of the parameters of weighting models, it seems natural to question if their training is necessary for attaining effective learned models. Indeed, if the proper setting of parameters is shown to be unnecessary, then the expensive training process could be omitted. Hence, from this arises the second research question that we tackle in this paper:

RQ 2. *Should the parameters of document weighting model features be trained before deployment within a learned model for effective retrieval?*

3.3 Fields and Weighting Models

Similar arguments arise concerning the use of weighting models based on different document representations (or fields)[4]. Firstly, documents can be represented in a semi-structured fashion, depending on where within the document the various terms occur. To illustrate this, consider Figure 2, which illustrates the different document representations used by both the LETOR v3.0 dataset and also applied in this work. While term occurrences in the document's main content (known as the body) are important, their occurrences in other fields including title and URL may also be important. We draw attention to the anchor text field, where the text associated with hyperlinks from incoming documents is added to the target document.

With respect to the role of document representations in weighting models, in early work, Katzer et al. [1982] found that different document representations gave similar retrieval effectiveness, but found different sets of relevant documents - known as the skimming effect by Vogt and Cottrell [1998]. This suggests that within a learned model, there is benefit in defining different query dependent weighting model features for each field f of a document d. For example, a linear combination

[4]We use the terms 'fields' and 'document representations' interchangeably.

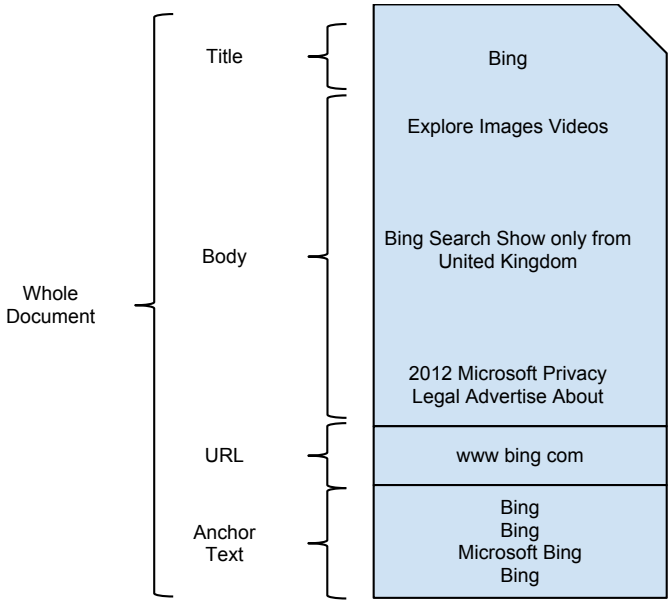

Fig. 2. Different document representations (fields) of a Web document.

of weighting model scores (as learned by a linear learning to rank technique) takes the following form:

$$score(d, Q) = \sum_{t \in Q} \sum_{f} w_f \cdot w(tf_f) \tag{3}$$

where each query term t occurs tf_f times in field f of documents d, and the importance of a term's occurrences in a field f is weighted by w_f. $w()$ is a function that scores the occurrences of a term within a document representation, and could be any typical document weighting model, such as BM25. We call the calculation of a weighting model for each field *single-field models*. Ogilvie and Callan [2003] as well as Kamps [2006] have both considered the combination of different single-field models using data fusion techniques.

However, Robertson et al. [2004] made a theoretical argument against the linear combination of weighting model scores for different document representations. They contrasted this approach to the linear weighted combination of the frequencies of query terms within each field:

$$score(d, Q) = \sum_{t \in Q} w(\sum_{f} w_f \cdot tf_f), \tag{4}$$

which we refer to as a *field-based* weighting model. Contrasting the two approaches, five different issues were identified with the combination of weighting model scores calculated separately on different fields:

(i) **nonlinear TF**: through a figure, Robertson et al. [2004] show that due to the non-linear nature of the various term frequency saturation functions, the

combination of term scores would result in an inflated overall score for a document than the combination of frequencies, as follows:

$$w(\sum_f w_f \cdot tf_f) \leq \sum_f w_f \cdot w(tf_f) \tag{5}$$

(ii) **equal field weights**: It was argued that if the field importance weights w_f were equal, then it was natural that the weighting should revert to the case of an unstructured document. However, for $w_f = 1$, in the combination of term weights rather than frequencies, this does not hold:

$$w(\sum_f w_f \cdot tf_f) \neq \sum_f w_f \cdot w(tf_f), w_f = 1 \forall f \tag{6}$$

(iii) **background statistics**: weighting models typically examine the importance of a term with respect to its occurrences in a corpus, known as background statistics. For a model computed on a single representation, this would naturally be its occurrences within that representation of the document. However, Robertson et al. [2004] argued that the background statistics for some types of fields (such as the title of documents) could be quite sparse in contrast to the corpus as a whole, leading to poorer estimation of term importance.

(iv) **number of parameters**: if weighting model scores are calculated for each document – as per Equation (3) – then the number of weighting model parameters that may require training is multiplied by the number of document representations. For instance, if the b and k_1 parameters of BM25 require training for invocation on three different fields, then including a weighting parameter w_f for their weighting linear combination, this would amount to 9 parameters for training (i.e. 3 parameters $\times$ 3 fields). In contrast, for the linear combination of field frequencies (Equation (4)), only the three w_f parameters were necessary in addition to b and k_1.

(v) **document length**: the final argument considered that document length normalisation was originally motivated in terms of the verbosity of a document. However, different fields have different length characteristics: terms in the title and URL fields normally occur once; anchor text may have many repeated terms. Robertson et al. [2004] questioned if normalisation should be performed on separate fields or not. Indeed, in the formulation of Equation (4), normalisation with respect to document length only occurs once for a document. Later, a revised approach also known as BM25F [Zaragoza et al. 2004] was proposed, as well as the similar PL2F [Macdonald et al. 2006]. Both of these models perform normalisation on a per-field basis, suggesting that length normalisation is better conducted in a per-field manner:

$$score(d, Q) = \sum_{t \in Q} w(\sum_f w_f \cdot tfn_f) \tag{7}$$

where tfn_f is the normalised term frequency with respect to the length of field f and the average length of the field for all documents, as calculated by

some term frequency normalisation function [Zaragoza et al. 2004; Macdonald et al. 2006][5].

After these theoretical arguments, Robertson et al. [2004] provided experiments validating that the employment of a field-based model which linearly combined the *frequency* of query term occurrences in different fields did exhibit superior performance compared to the linear combination of weighting model scores from separate fields (single-field models). The effectiveness of BM25F and PL2F have also been shown in the literature [Macdonald et al. 2006; Plachouras 2006; Zaragoza et al. 2004].

We contrast these stated theoretical and empirical arguments from the literature with the practices exhibited in existing learning to rank datasets. Indeed, the LETOR v3.0 dataset deploys various weighting models, each computed on the whole document, as well as single-field model features on four constituent document representations, namely body, title, URL, and the anchor text of incoming hyperlinks (for instance, see feature IDs 21-25 in Table I, which calculate BM25 on five different document representations). However, a field-based weighting model such as BM25F [Zaragoza et al. 2004] is not deployed. The same situation is observed with the MSLR-WEB10K and MSLR-WEB30K datasets.

There are theoretical and empirical arguments advocating field-based weighting models, as well as against the *linear* combination of weighting models for different representations. Yet, at least some learning to rank techniques deploy such linear combinations of features (e.g. AFS [Metzler 2007b], Adarank [Xu and Li 2007] and RankSVM [Joachims 2002]), which hence may be unsuitable for use with multiple weighting model features for different representations. On the other hand, it is unclear if arguments against the combination of such single-field model features apply for other non-linear learning to rank techniques, such as those based on regression trees (e.g. LambdaMART). Indeed, in the latter case, as the outcome of a single regression tree is dependent on the feature values, but not actually dependent on any form of linear combination of those feature values, then the argument against combining single-field model features may no longer be appropriate.

This gives rise to our third research question, where we examine the use of weighting models for fields as features by learning to rank techniques. In particular, we compare field-based models – where the term frequency from different document representation is linearly combined into one weighting model features – with single-field model features – where weighting model scores are obtained separately for each document representation:

RQ 3. *Are field-based or single-field weighting models more effective as features?*

In the remainder of this paper, we experiment to address these three research questions, using standard datasets within a variety of settings. In the next section, we detail the experimental setup, while results and conclusions follow thereafter.

[5]It is of note that other per-field normalisation field-based models can be generated using the generic outline of Equation (7), by alteration of the normalisation function for each field that produced tfn, or the weighting function $w()$ - other effective functions are provided in [He 2007].

4. EXPERIMENTAL SETUP

Our experiments are conducted in a Web search setting, using features that are often applied for ranking in Web corpora. In the previous section, we highlighted many weighting model features that may be used in an effective learned Web search model, such as field-based weighting models (e.g. BM25F) and single-field weighting models (e.g. BM25 computed separately on each field). However, computing all of these features for every document matching more than one query term would result in unnecessary inefficiencies at retrieval time. Instead, Section 4.1 describes the framework that we use for computing such query dependent features. Following this, Section 4.2 details the dataset and features we deploy in our experiments, while Section 4.3 describes the used learning to rank techniques.

4.1 Fat Framework for Calculating Query Dependent Features

Calculating all query dependent features for every document that matches one or more query terms would be extremely inefficient - indeed, a query term may match millions of documents, but only a few thousand are needed for the sample to be reranked by the learned model. Instead, it is natural to only compute these features for the documents that actually make the sample of size k. However, the information to calculate these features (typically term frequencies) requires further access to the inverted file posting lists. Moreover, due to their compression, inverted file posting lists are always accessed in a stream fashion, which makes random access for only the documents within the sample impossible. One alternative would be to rescan the posting lists for the documents in the sample. Nevertheless, as the sample of documents is typically several orders of magnitude smaller than the length of the posting lists, this would be particularly slow. Another alternative is to assume that the entire posting lists for all query terms can be retained in memory - indeed this is the approach taken by the Indri platform[6], however for large corpora with long posting lists this is not practical.

Instead, in this work, we use an approach that allows the additional document weighting model features to be calculated only when the final sample has been finalised. Firstly, in the manner described by Broder et al. [2003] and outlined in Figure 3, we assume that the posting lists for each query term are accessed using an iterator pattern, which allows access to each posting in ascending docid order. In particular, `next()` moves the iterator to the term's next posting, while `next(id)` allows skipping [Broder et al. 2003; Moffat and Zobel 1996] onto the next document with docid greater than that specified. Indeed, `next(id)` facilitates advanced document-at-a-time (DAAT) dynamic pruning strategies such as WAND [Broder et al. 2003]. The `frequency()` and `docLength()` methods permit access to the statistics of the term's occurrence in the document represented by the current posting.

Building upon the general pattern shown in Figure 3, we add a `clone()` method, which returns a *copy* of the current posting, free of the underlying iterator. Note that we use an object to represent each posting, as depending on the used indexing configuration, each posting may contain additional information, such as field fre-

[6] `http://www.lemurproject.org/indri/`

Posting

return the frequency for the current posting
`int frequency()`
return the document length for the current posting
`int docLength()`

Posting Iterator extends Posting

move to the next posting and return its docid
`int next()`
move to the next posting with docid ≥ target, return posting's actual docid
`int next(target)`

return a copy of the current posting, free of the current iterator `Posting clone()`

Fig. 3. Interfaces permitting access to a posting, and the iterator on a term's posting list. The clone() method is an important addition that makes the fat framework possible, by keeping a copy of the current posting free of the underlying iterator on the term's posting list in the inverted index.

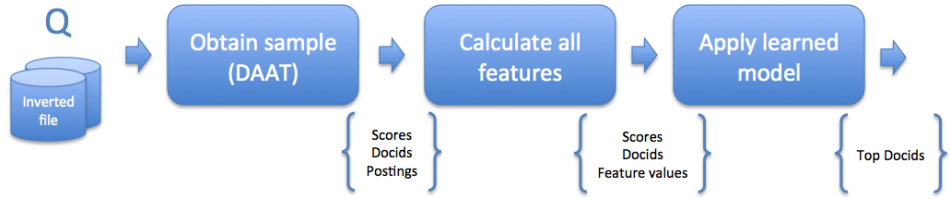

Fig. 4. Retrieval stages within the fat framework.

quencies, occurrence positions and other payload information, that also need to be copied.

Figure 4 shows the stages of retrieval (corresponding to steps 4-6 from Section 2) and the flow of information between them. In particular, the first-stage retrieval identifies the sample, during which documents are scored in the DAAT fashion. Whenever a document scores high enough to be a candidate to be retrieved in the result set of k sample documents, all postings of the current document matching the query terms are saved along with the document's score to the result set. Once the result set has k documents, each newly retrieved document exceeding the threshold (the score of the current k lowest scored document) is added to the result set, while the k-th ranked document and its saved postings are expelled and discarded.

After the first-stage retrieval to identify the sample, the postings for all k retrieved sample documents are available to calculate all other weighting model features. In particular, we say that the result set is *fat*, as it contains as much information as possible for the top k documents for the current query. Hence, there is no need to access the inverted file to calculate more weighting model features for these documents. Moreover, the amount of memory required to store the postings (or calculated features) for each query is fixed to an amount linear with the size of the sample (k), instead of the number of documents matching any query term, which will vary for each query, and may exceed the available memory for large disk-based indices. In practice, a fat result set must also retain other statistics to allow weighting model features to be calculated, listed below:

—**collection statistics**: e.g. the number of documents and the number of tokens in the collection, such that document length normalisation can occur. This can also include the number of tokens in each field, to permit per-field normalisation in field-based weighting models.

—**term statistics**: e.g. the number of documents containing the term, so that background models/IDF can be calculated. Similarly to collection statistics, for the purposes of field-based weighting models, this can also include the number of occurrences in each field.

—**query statistics**: e.g. number of times each query term occurs in the query, such that query terms re-occurring in long queries can be appropriately weighted.

Finally, once the additional features are calculated, the third stage calculates the final document ranking by using a previously learned model combining the features. While we use the fat framework to facilitate experiments within this paper, it has advantages for both the deployment of IR systems and for conducting experimental IR, as detailed below:

—*Using the Fat Framework for Deployed IR systems:* The fat framework can be used as the basis of a search engine deploying many features in a learned model. It is of note that only a DAAT retrieval strategy can be used in the first stage retrieval to identify the sample. Indeed, as all postings for a given document are processed at the same time, the postings for a document that will not make the k documents in the sample are not required. In contrast, for term-at-a-time retrieval strategies (TAAT), all postings for a given query term would have to be retained in memory until the top k was decided. This may outstrip the available memory. In contrast, with the fat framework, due to its DAAT nature, only space for the postings of k documents is required.
The fat framework can be seen as orthogonal to that of Cambazoglu et al. [2010]. Their approach focuses on the pruning of documents that cannot make the retrieved set when applying the learned model. Firstly, they assume that all features are already computed and held in memory. Then they propose that the cost of scoring a document can be reduced by early terminating the accumulation of the final score from the constituent regression trees of the learned model (in a manner somewhat similar to the DAAT MaxScore dynamic pruning strategy [Turtle and Flood 1995]). Depending on the index ordering, different variants of the approach of Cambazoglu et al. [2010] can result in effectiveness degradation. Their approach could be applied in conjunction with the fat framework for regression tree-based learned models during the application of the learned model.

—*Using the Fat Framework for Experimental IR:* As discussed above, the primary advantage of a fat result set is that additional query dependent document features can be calculated without resorting to a second access of the inverted file posting lists. A fat result set also has several other benefits when developing and experimenting with learned ranking models. For instance, the fat result sets for a training query set can be saved to disk, such that more features can be calculated without further access to the inverted file. Moreover, in contrast to an inverted file, the fat result sets for a substantial set of queries can easily fit in memory, re-

	CW09B	MQ2007 (LETOR v4.0)	MQ2008 (LETOR v4.0)
Training	88	1014	470
Validation	30	339	157
Testing	30	339	157

Table II. For each query set, the number of queries for training, validation and testing in each of 5 folds. The total number of queries are 150 (CW09B), 1692 (MQ2007) and 784 (MQ2008).

Representation	Tokens	Avg. Length
CW09B		
Entire Document	52,031,310,320	1036.0
Body	38,314,153,346	762.9
URL	373,922,803	7.4
Title	396,214,008	7.9
Anchor Text	12,947,022,766	257.8
GOV2 (LETOR v4.0)		
Entire Document	19,910,298,659	789.9
Body	19,215,162,601	762.3
URL	469,974,906	18.6
Title	138,172,107	5.5
Anchor Text	86,989,045	3.5

Table III. Statistics across four fields of the 50M document CW09B corpus (as indexed by Terrier), and the 25M document GOV2 corpus (as provided by LETOR v4.0).

gardless of the number of documents in the underlying index. This permits the efficient training of parameters of features, without repeated posting list traversals.

4.2 Datasets and Features

Our experiments are performed using two different corpora, namely the TREC 'category-B' ClueWeb09 Web crawl (CW09B), and the Million Query 2007 & 2008 track query sets from the LETOR v4.0 dataset, which are based on the TREC GOV2 corpus.

The CW09B corpus comprises 50 million English Web documents and is aimed to represent the first tier of a commercial search engine index [Callan et al. 2009]. With this corpus, we use queries and relevance assessments from three tasks of the TREC Web track, 2009, 2010 and 2011 [Clarke et al. 2010; 2011; 2012]. In particular, all 150 queries are mixed within a 5-fold cross validation, where in each fold, there are 3 parts training queries, 1 part validation queries and 1 part test queries. The 5-fold cross validation permits sufficient training and validation data while smoothing the differences in the natures of the three tasks (e.g. TREC 2011 targets less frequent, *"more obscure topics [queries]"* [Clarke et al. 2012]).

On the other hand, the GOV2 corpus comprises a Web crawl of 25 million documents from the `.gov` domain. The LETOR v4.0 dataset provides standard queries and features for 1692 and 784 queries from the TREC 2007 and 2008 Million Query tracks, respectively. Importantly for our work, the dataset also provides low-level information, namely document-level, term-level and collection-level statistics, which permit new query dependent features to be calculated. Each query set is split into 5 folds, each with training, validation and testing queries. Salient statistics of the three query sets are shown in Table II.

We implement the fat framework on version 3.5 of the Terrier IR platform [Macdonald et al. 2012]. Hence, using Terrier, we index the documents of the CW09B corpus, while considering the body, the title, the URL, and the anchor text from the incoming hyperlinks to be separate fields of each document. For LETOR v4.0, we use the fat framework to create postings from the provided low-level statistics instead of a traditional inverted index. Table III reports the total number of indexed tokens and the average length of each field for both CW09B and LETOR.

For CW09B, at retrieval time, for each query, we use a light version of Porter's English stemmer, and rank 5000 documents within the sample, following Craswell et al. [2010]. We experiment with two document weighting models for the sample, namely BM25 [Robertson et al. 1992] and PL2 [Amati 2003] from the Divergence From Randomness framework, without considering anchor text, as this results in the samples with the highest recall of relevant documents[7]. For the LETOR query sets, we are limited to the proscribed sampling method, where a mean of 41 documents are sampled using BM25.

On both datasets, for all sample documents, we calculate additional query dependent (QD) and query independent (QI) features. The deployed features are often used in the learning to rank literature [Qin et al. 2009]. In particular, we use the same QD features for both CW09B and LETOR query sets, however, for LETOR we are limited to the QI features provided by the dataset. All deployed features are summarised in Table IV and further sub-divided to support the research questions:

QI - Features from hyperlink analysis, URL length, and document quality.

QD:WMWD - Multiple standard weighting models computed on whole documents (the whole document contains terms occurring in all fields, as per Figure 2).

QD:WMSF - A standard weighting model calculated individually on each "single field" of the document. In particular, we use PL2 or BM25.

QD:WMFB - A weighting model that is field-based, where term frequencies are combined rather than the weighting model scores. We deploy BM25F [Zaragoza et al. 2004] and PL2F [Macdonald et al. 2006], which both independently normalise and weight the term frequencies from each field.

Two weighting models form the root of many features within this paper, namely BM25 and PL2, with their corresponding field-based variants, i.e. BM25F [Zaragoza et al. 2004] and PL2F [Macdonald et al. 2006]. In BM25, the relevance score of a document d for a query Q is given by:

$$score(d, Q) = \sum_{t \in Q} w^{(1)} \frac{(k_1 + 1)tfn}{k_1 + 1 + tfn} \frac{(k_3 + 1)qtf}{k_3 + qtf} \tag{8}$$

where qtf is the frequency of the query term t in the query Q; k_1 and k_3 are parameters, for which the default setting is $k_1 = 1.2$ and $k_3 = 1000$ [Robertson et al. 1995]; $w^{(1)}$ is the *idf* factor, given by $w^{(1)} = \log \frac{N - N_t + 0.5}{N_t + 0.5}$ where N is the number of documents in the collection, and N_t is the number of documents containing the

[7]A sample size of 5000 documents and use of a representation without anchor text follows from the recommandations in [Macdonald et al. 2012].

Class	Features	Total
	Weighting Models on Whole Documents	
QD:WMWD	BM25 [Robertson et al. 1992] Dirichlet [Zhai and Lafferty 2001] PL2 [Amati 2003] DPH [Amati et al. 2008] No. of matching query terms [Qin et al. 2009]	4
	Weighting Models on Single Fields	
QD:WMSF	PL2 BM25	4
	Field-based weighting model	
QD:WMFB	BM25F [Zaragoza et al. 2004] or PL2F [Macdonald et al. 2006]	2
	Query Independent	
QI (CW09B)	Absorbing [Plachouras et al. 2005] Inlinks PageRank Edgerecip [Becchetti et al. 2006] URL Length, type etc Field lengths Content quality [Bendersky et al. 2011] Spam likelihood [Cormack et al. 2011]	23
QI (LETOR)	PageRank Inlinks Outlinks URL length, etc Number of child pages Field lengths	8

Table IV. Features deployed on ranking documents. The query independent features vary in the CW09B and LETOR query sets.

query term. The normalised term frequency tfn is given by:

$$tfn = \frac{tf}{(1+b) + b \cdot \frac{l}{avg\_l}}, (0 \leq b \leq 1) \tag{9}$$

where tf is the term frequency of the term t in document d. b is the term frequency normalisation hyper-parameter, for which the default setting is $b = 0.75$ [Robertson et al. 1995]. l is the document length in tokens and avg_l is the average document length in the collection.

In contrast to BM25, for BM25F instead of calculating the normalised term frequency tfn using Equation (9), tfn is instead obtained by normalising the term frequency tf_f from each field f (body, title, anchor text, URL) separately:

$$tfn = \sum_f w_f \cdot \frac{tf_f}{(1-b_f) + b_f \cdot \frac{l_f}{avg\_l_f}}, (0 \leq b_f \leq 1) \tag{10}$$

where tf_f is the term frequency of term t in field f of document d, l_f is the length in tokens of field f in document d, and avg_l_f is the average length of f in all documents of the collection. The normalisation applied to terms from field f can be controlled by the field hyper-parameter, b_f, while the contribution of the field is controlled by the weight w_f.

In the PL2 weighting model [Amati 2003], the importance of a term occurring in a document is obtained by measuring its divergence from the expectation given its occurrence in the corpus. Hence, PL2 scores term occurrences in documents as follows:

$$score(d,Q) = \sum_{t \in Q} qtw \cdot \frac{1}{tfn+1}\big(tfn \cdot \log_2 \frac{tfn}{\lambda} + (\lambda - tfn) \cdot \log_2 e + 0.5 \cdot \log_2(2\pi \cdot tfn)\big) \quad (11)$$

where λ is the mean and variance of a Poisson distribution, given by $\lambda = F/N$. F is the frequency of terms t in the entire corpus. In the Divergence From Randomness (DFR) framework, the query term weight qtw is given by qtf/qtf_{max}. qtf is the query term frequency. qtf_{max} is the maximum query term frequency among the query terms.

To accommodate document length variations, the normalised term frequency tfn is given by the so-called Normalisation 2 from the DFR framework:

$$tfn = tf \cdot \log_2(1 + c \cdot \frac{avg\_l}{l}), (c > 0) \quad (12)$$

where tf is the actual term frequency of the term t in document d and l is the length of the document in tokens. avg_l is the average document length in the whole collection ($avg\_l = \frac{token_c}{N}$). c is the hyper-parameter that controls the normalisation applied to the term frequency with respect to the document length. The default value is $c = 1.0$ [Amati 2003].

In the PL2F model, the document length normalisation step is altered to take a more fine-grained account of the distribution of query term occurrences in different fields. The so-called Normalisation 2 (Equation (12)) is replaced with *Normalisation 2F* [Macdonald et al. 2006], so that the normalised term frequency tfn corresponds to the weighted sum of the normalised term frequencies tf_f for each used field f:

$$tfn = \sum_f \left(w_f \cdot tf_f \cdot \log_2(1 + c_f \cdot \frac{avg\_l_f}{l_f}) \right), (c_f > 0) \quad (13)$$

where c_f is a hyper-parameter for each field controlling the term frequency normalisation, and the contribution of the field is controlled by the weight w_f. Together, c_f and w_f control how much impact term occurrences in a field have on the final ranking of documents. Again, tf_f is the term frequency of term t in field f of document d, l_f is the number of tokens in field f of the document, while avg_l_f is the average length of field f in all documents, counted in tokens. Having defined Normalisation 2F, the PL2 model (Equation (11)) can be extended to PL2F by using Normalisation 2F (Equation (13)) to calculate tfn.

For both BM25F and PL2F, the proper setting of the normalisation parameters (b_f and c_f) as well as the field weights (w_f) depends on the distribution of lengths observed in each fields, and their importance in effective retrieval. Hence, in contrast to BM25's b and PL2's c parameters, there are no default recommended settings. Instead, as detailed below, we train these parameters to maximise performance on a training set of queries.

The other weighting models from Table IV are language modelling using Dirichlet smoothing [Zhai and Lafferty 2001], DPH [Amati et al. 2008] and the number of matching query terms [Qin et al. 2009]. In Dirichlet, the score of a document is rank equivalent under log transform to the smoothed probability query likelihood $P(d|Q)$ using a Dirichlet prior:

$$P(Q|d) = \prod_t \lambda P(t|d) + (1-\lambda)P(t) \tag{14}$$

where $P(t|d)$ is the maximum likelihood of the occurrence of query term t in document d, and the $P(t)$ is likelihood of t occurring in the whole corpus. Following Zhai and Lafferty [2001], $\lambda = \frac{\mu}{l+\mu}$.

In DPH [Amati et al. 2008], the score of a document is obtained as follows:

$$\begin{aligned} score(d,Q) \;=\; & \sum_{t \in Q} \frac{qtw(1-\frac{tf}{l})^2}{tf+1} \cdot \big(tf \cdot \log_2(tf \cdot \frac{avg\_l}{l}\frac{N}{F})\big) \\ & +0.5 \cdot \log_2(2\pi \cdot tf \cdot (1-\frac{tf}{l})) \end{aligned} \tag{15}$$

Finally, the use of the number of matching query terms as a query dependent feature allows a learner to estimate if the document would have been retrieved by a conjunctive retrieval approach:

$$score(d,Q) = \sum_{t \in Q} \begin{cases} 1 & \text{if } tf > 0, \\ 0 & \text{otherwise.} \end{cases}$$

4.3 Learners

Various learning to rank techniques in the literature fall into one of three categories, namely pointwise, pairwise and listwise [Liu 2009]. Purely pointwise approaches are rarely used, as they do not account for the relative importance of documents. Instead, we deploy two listwise learners and one pairwise learner, which differ in the form of their learned models:

—Automatic Feature Selection (AFS) [Metzler 2007b] obtains a weight for the linear combination of the most effective feature at each iteration, which is then added to the set of features already selected in the previous iteration(s). In our implementation, we use simulated annealing [Kirkpatrick et al. 1983] to find the combination weight for each feature that maximises NDCG@1000[8]. Note that such weights are obtained one by one, with no revisiting of the weights of the already selected features. When validation data is used, the model of the highest performing iteration as measured using the same evaluation measure on the validation data is chosen (in this manner, the validation data is used to determine the correct number of AFS iterations, as suggested by Liu [2009]).

[8]While a rank cutoff of 1000 increases the informativeness of the measure to additional swaps in the ranking, it will likely not impact the final effectiveness of the learned model [Macdonald et al. 2012].

—RankSVM [Joachims 2002] - a pairwise technique, as implemented by the svm_{rank} package[9]. To determine the C_{rank} parameter of RankSVM, which affects the trade-off between training error and margin, we pick the highest performing $C_{rank} = \{0.01, 0.05, 0.25, 1.25, 6.25, 31.25\}$, based on NDCG@1000 as computed on the validation set. In the default svm_{rank} configuration, a linear kernel is used, such that the resulting model is a linear combination of features.

—LambdaMART [Wu et al. 2008] deploys boosted regression trees internally, where the learning of the trees considers NDCG to obtain the gradient of the surrogate loss function between pairs of documents. We use the implementation of the Jforests open source package [Ganjisaffar et al. 2011][10]. A LambdaMART approach was the winning entry in the 2011 Yahoo! learning to rank challenge [Chappelle and Chang 2011]. The model of the highest performing iteration on the validation data is chosen.

Finally, for training the parameters of the weighting models (namely b in Equation (9) for BM25, μ for Dirichlet and c in Equation (12) for PL2, as well as b_f and w_f in Equation (10) for BM25F and c_f and w_f in Equation (13) for PL2F), we use simulated annealing to maximise NDCG@1000. This training occurs independently of the learning to rank approach used to combine these features. When untrained, these parameters are left at their default settings from the literature ($b = 0.75$ [Robertson et al. 1995], $c = 1$ [Amati 2003] and $\mu = 2500$ [Zhai and Lafferty 2001]).

In the following, we report the results and analysis for each of our three research questions in Sections 5, 6 and 7, respectively. Each section consists of experiments, analysis and summary.

5. MULTIPLE DOCUMENT WEIGHTING MODEL FEATURES

Recall that our first research question addresses whether the use of multiple weighting models as query dependent features leads to an increased effectiveness. In the following, Section 5.1 details our experiments to address the research question across the CW09B and LETOR v4.0 query sets. In Section 5.2, we study the different weighting model features used, including their usefulness and the correlation of their generated rankings to the sample model, so as to understand whether they can bring new evidence to improve the effectiveness of the learned model. We summarise our findings for this research question in Section 5.3.

5.1 Experiments

In this section, we aim to determine if deploying multiple weighting models computed on the whole document representation can result in increased effectiveness over the weighting model used to obtain the sample alone. To do so, we control both the presence of query independent features within the learned model, and the choice of learner from those described in Section 4.3. In addition, for the CW09B query set only, we control the choice of sample model, namely BM25 or PL2 – indeed, recall from Section 4.2 that for the LETOR v4.0 query sets, we can only use

[9] http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html
[10] http://code.google.com/p/jforests/

Sample	Features QD	QI	Total	# Selected	NDCG@20	NDCG@3	ERR@20
AFS							
BM25	✗	✗	1	1	0.2056	0.1932	0.0938
BM25	WMWD	✗	6	4.0	0.2392≫	0.2381>	0.1181≫
BM25	✗	✔	24	10.0	0.2606	0.2655	0.1326
BM25	WMWD	✔	29	12.8	0.2791=	0.2960=	0.1453=
PL2	✗	✗	1	1	0.1762	0.1575	0.0795
PL2	WMWD	✗	6	4.2	0.2081>	0.2109>	0.1040≫
PL2	✗	✔	24	10.2	0.2418	0.2335	0.1168
PL2	WMWD	✔	29	12.4	0.2665=	0.2809>	0.1389≫
RankSVM							
BM25	✗	✗	1	1	0.2056	0.1932	0.0938
BM25	WMWD	✗	6	-	0.2247=	0.2088=	0.1071>
BM25	✗	✔	24	-	0.2771	0.2775	0.1361
BM25	WMWD	✔	29	-	0.2925=	0.3193≫	0.1520≫
PL2	✗	✗	1	1	0.1762	0.1575	0.0795
PL2	WMWD	✗	6	-	0.2105≫	0.1976>	0.1006≫
PL2	✗	✔	24	-	0.2483	0.2667	0.1246
PL2	WMWD	✔	29	-	0.2821≫	0.3127>	0.1484≫
LambdaMART							
BM25	✗	✗	1	1	0.2056	0.1932	0.0938
BM25	WMWD	✗	6	6.0	0.2266=	0.2018=	0.1074=
BM25	✗	✔	24	20.2	0.2620	0.2532	0.1277
BM25	WMWD	✔	29	22.6	0.2735=	0.2790=	0.1369=
PL2	✗	✗	1	1	0.1762	0.1575	0.0795
PL2	WMWD	✗	6	6.0	0.2110≫	0.2028>	0.1004≫
PL2	✗	✔	24	20.4	0.2044	0.2166	0.1118
PL2	WMWD	✔	29	23.4	0.2844≫	0.3070≫	0.1467≫

Table V. Results on the CW09Q query set comparing the use of multiple (query dependent) weighting models as features in a learned model. NDCG@20, NDCG@3 and ERR@20 across 5 folds are reported, as well as the mean number of selected features (RankSVM always selects all features, and hence the numbers of selected features are omitted for this learner).

the pre-determined samples for each query. Moreover, in these experiments, the parameters of the weighting model features remain untrained using their default settings as listed in Section 4.3 – instead, we address the training of the parameters later, in Section 6. Note that the fat framework is very useful in conducting this experiment, as a fat result set can easily have any number of additional features calculated without resorting to further expensive inverted file disk accesses.

For the CW09B and LETOR query sets, Tables V & VI respectively report the performance across various settings when varying the presence of multiple document weighting models computed on the whole document representation within the learned model. Mean performances are reported across 5 folds of the test topics, using the appropriate evaluation measures. In particular, for the CW09B query set, we report NDCG@20 and ERR@20 (which were the official TREC measures [Clarke et al. 2011]). For the LETOR query sets, we report NDCG@10 and MAP@10. Moreover, for both query sets, we additionally report NDCG@3, to reflect a user's perspective on the top ranked documents. For a given row of each table, to show whether the addition of query dependent features benefits effectiveness, statistically significant increases (decreases) from the row above in the tables,

Query set	Features QD	QI	Total	# Selected	NDCG@10	NDCG@3	MAP@10
AFS							
MQ2007	✗	✗	1	1	0.3036	0.1674	0.1968
MQ2007	WMWD	✗	5	4.6	0.3651≫	0.2073≫	0.2519≫
MQ2007	✗	✔	9	4.4	0.3148	0.1698	0.2040
MQ2007	WMWD	✔	13	7.2	0.3641≫	0.2066≫	0.2508≫
MQ2008	✗	✗	1	1	0.4635	0.3179	0.3932
MQ2008	WMWD	✗	5	2.6	0.4977≫	0.3503≫	0.4290≫
MQ2008	✗	✔	9	2.2	0.4602	0.3073	0.3888
MQ2008	WMWD	✔	13	5.0	0.4961≫	0.3481≫	0.4276≫
RankSVM							
MQ2007	✗	✗	1	1	0.3036	0.1674	0.1968
MQ2007	WMWD	✗	5	-	0.3583≫	0.2020≫	0.2456≫
MQ2007	✗	✔	9	-	0.3093	0.1682	0.1999
MQ2007	WMWD	✔	13	-	0.3562≫	0.1992≫	0.2431≫
MQ2008	✗	✗	1	1	0.4635	0.3179	0.3932
MQ2008	WMWD	✗	5	-	0.5033≫	0.3600≫	0.4382≫
MQ2008	✗	✔	9	-	0.4618	0.3118	0.3891
MQ2008	WMWD	✔	13	-	0.5003≫	0.3543≫	0.4338≫
LambdaMART							
MQ2007	✗	✗	1	1	0.3036	0.1674	0.1968
MQ2007	WMWD	✗	5	5.0	0.3611≫	0.2060≫	0.2481≫
MQ2007	✗	✔	9	8.0	0.3223	0.1747	0.2089
MQ2007	WMWD	✔	13	12.0	0.3662≫	0.2076≫	0.2505≫
MQ2008	✗	✗	1	1	0.4635	0.3179	0.3932
MQ2008	WMWD	✗	5	5.0	0.4919≫	0.3457≫	0.4244≫
MQ2008	✗	✔	9	8.0	0.4714	0.3217	0.3985
MQ2008	WMWD	✔	13	12.0	0.4983≫	0.3488≫	0.4302≫

Table VI. Results on the LETOR query sets comparing the use of multiple (query dependent) weighting models as features in a learned model. NDCG@10 and MAP@10 across 5 folds are reported, as well as the mean number of selected features (RankSVM always selects all features, and hence the numbers of selected features are omitted for this learner).

Feature Set						
QI	✗	✔	✗	✔	✗	✔
CW09B	Mean NDCG@20		Mean NDCG@3		Mean ERR@20	
(sample)	0.1909	0.2644	0.1753	0.2757	0.0866	0.1348
WMWD	**0.2200** +15.2%	**0.2797** +5.7%	**0.2100** +19.7%	**0.2992** +8.5%	**0.1063** +22.7%	**0.1447** +6.4%
LETOR	Mean NDCG@10		Mean NDCG@3		Mean MAP@10	
(sample)	0.3836	0.4101	0.2426	0.2598	0.2950	0.3188
WMWD	**0.4296** +12.0%	**0.4302** +4.9%	**0.2785** +14.7%	**0.2774** +6.8%	**0.3395** +15.1%	**0.3393** +6.4%

Table VII. Summary table for Table V & VI, detailing the mean effectiveness for different feature sets, across different samples and learners.

according to the paired t-test are denoted by $>$ ($<$) for $p < 0.05$ and $\gg$ for $p < 0.01$. $p \geq 0.05$ is denoted by $=$. Finally, the number of features available, and the mean number of features selected by the AFS and LambdaMART learners across 5 folds are shown (RankSVM always selects all features, and hence the numbers of selected features are omitted for this learner).

For example, the second row of Table V is explained as follows: the sample is obtained using BM25; WMWD denotes that in addition to the sample, the 5

whole document weighting model features are deployed (BM25, Dirichlet language modelling, PL2, DPH, and the number of matching query terms); In addition, there are no query independent features; Of the 6 total features, on average across the 5 folds, 4 features have been selected in the learned model obtained from AFS; The resulting mean retrieval performances across the test topics of each fold are 0.2392 (NDCG@20), 0.2381 (NDCG@3) and 0.1181 (ERR@20) - all of which represent significant increases over the line above, which does not deploy multiple query dependent features.

To permit easy analysis of Tables V & VI across the multiple dimensions, summary Table VII reports the mean effectiveness for different feature sets (with or without WMWD), across different samples and learners (AFS, RankSVM and LambdaMART). For instance, in the first row of Table VII, mean NDCG@20 of 0.1909 is the mean of the NDCG@20 of both the BM25 and PL2 samples (first row for each learner group) in Table V.

Our analysis of Table V & VI always consider pairs of rows, without and with multiple whole document weighting model (WMWD) features. In general, on analysing these tables, we note marked increases on adding multiple weighting models as features. Indeed, 23 out of a total 36 cases in Table V exhibit statistically significant increases, while all 36 cases exhibit significant improvements in Table VI. These results can be observed across different learners and samples, as shown by the increases observed in summary Table VII. Indeed, for the CW09B query set (Table V), significant increases are observed for each learner: AFS (8 out of 12 cases); LambdaMART (6 cases); RankSVM (9 cases). Moreover, across all query sets – CW09B, MQ2007 and MQ2008 – the number of selected features always increases when adding multiple weighting models into the feature set, illustrating the perceived usefulness by the learners of the multiple weighting model features, and their resulting positive impact on the effectiveness of the resulting learned models.

The learned models that encapsulate query independent features perform considerably better, showing the value of the deployed QI features for Web search. Adding multiple weighting models (WMWD) to QI features always results in increases in performance. In particular, comparing the QI vs. WMWD+QI for the CW09B query set in Table V, the increases are statistically significant in 10 out of 18 cases. On the other hand, comparing sample vs. WMWD, 13 out of 18 significant increases are observed, suggesting that the WMWD feature set is more useful when QI features are not present. Similarly, for the LETOR query sets (Table VI), the improvements brought by multiple weighting models is less marked when QI features are present. Indeed, this result can also be observed for all query sets in Table VII, where the margin of relative improvement brought by the WMWD features is always less marked when QI features are already present. This suggests that the benefit of the multiple weighting model features is less important in the presence of other useful features.

Finally, we contrast the results for when BM25 and PL2 samples are used for the CW09B query set (Table V)[11]. In general, we note that while performance using the sample based on the PL2 weighting model are slightly lower, the observations are consistent across the two weighting models. The difference in performance can

[11]Recall that the LETOR v4.0 query set has a fixed sample.

be easily attributed to the slightly lower recall of the PL2 samples (indeed, on further inspection of the samples, we find that the PL2 sample identified 63% of the relevant documents vs. 65% for BM25).

5.2 Analysis

Our findings above suggest that multiple query dependent features resulting in effective learned models, particularly when the QI features are not present. In the following, we aim to determine which query dependent features bring the most benefit in addition to the sample weighting model, contrasted with their similarity to the sample. In doing so, we bring an explanation as to why multiple document weighting model features are beneficial to the effectiveness of learned models. In particular, we postulate that the usefulness of the different weighting model features depends on how much they add to the evidence of relevance provided by the sample weighting model. For instance, Vogt and Cottrell [1998] enumerated the chorus, skimming and dark horse effects as possible reasons for the effectiveness of the data fusion combination of multiple retrieval approaches. In the following, we describe a method to characterise features inspired by these three effects, allowing us to understand how a given weighting model feature contributes towards an effective learned model.

In general, a chorus effect has been shown to be the most useful, as documents retrieved by multiple retrieval approaches are more likely to be relevant [Lee 1997]. In contrast, as noted in Section 3, the use of the sample in learning to rank ensures that all features retrieve exactly the same documents. To account for this in our approach, we ascertain if a feature ranks highly the same relevant documents as the weighting model used to create the sample (the *sample feature*). Indeed, features that do rank highly the same relevant documents as the sample are more likely to be exhibit a chorus effect.

To this end, we propose a 'top-heavy' set-based measure based on Dice's coefficient, which measures the similarity of the relevant documents ranked in the sample for two features F_1 and F_2. In particular, we define Dice Top (DT) as follows:

$$DT(F_1, F_2) = \frac{1}{Z} \sum_{i=1}^{N} Dice(F_1^i \cap R, F_2^i \cap R) \tag{16}$$

where F_1 is a ranking of sample documents by feature F_1, and F_1^i is the subset within ranks $1..i$. R denotes the set of relevant documents for the query that are present in the sample (and by extension in F_1 and F_2), while N is the size of the sample (e.g. $N = 5000$ for the CW09B query set, as per Section 4.2). Dice's coefficient for two sets X and Y is defined as $Dice(X, Y) = \frac{2|X \cap Y|}{|X|+|Y|}$ [van Rijsbergen 1979]. We normalise by Z, defined as N minus the number of top-ranked positions for which no relevant document appears for both features:

$$Z = N - \min(f_1, f_2) + 1 \tag{17}$$

where f_1 (respectively, f_2) is the (1-based) position of the first relevant document ranked by F_1 (resp., F_2). Note that the DT measure is 'top-heavy' - i.e. it primarily focuses on the top of the document ranking - as it considers $N-i$ times the similarity of documents at rank i. Moreover, $\frac{N+|R|}{N^2} \leq DT(F_1, F_2) \leq 1$, where 1 represents

the same ranking that ranks the relevant documents in the same order. Indeed, by using DT, we can ascertain the similarity of two different weighting model features at ranking highly the same relevant documents.

In addition to the DT similarity of a feature with respect to the sample feature, we also consider the effectiveness of a feature at ranking the sample documents without the presence of any other features (independent effectiveness). Indeed, a feature need not be effective in independence - e.g. a 'weak' feature such as URL length is only useful when combined with a stronger, query dependent feature. In the following, we postulate that such weaker features are more likely to generate skimming or dark horse effects, by only ranking highly some relevant documents some of the time.

Indeed, by making use of the DT and independent effectiveness measures, we are able to characterise the features that improve the retrieval effectiveness of a learned model, which is based upon a sample of reasonable effectiveness. In particular, we extend the chorus, skimming and dark horse definitions of Vogt and Cottrell [1998] as follows:

chorus effect: a feature generates a chorus effect if it is independently effective, retrieves the same relevant documents as the sample (i.e. has a high DT), yet still improves performance when added to the learned model. Indeed, by scoring highly the same relevant documents, it is providing reinforcing evidence to the learned model of the likely relevance of these documents.

skimming effect: a feature generates a skimming effect if it scores highly different relevant documents than a chorus feature. In doing so, it will have a lower DT, and may also have only a moderate independent effectiveness.

dark horse effect: a feature generates a dark horse effect if it has low independent effectiveness, as well as a lower DT, but can still improve the effectiveness of the learned model.

Of course, features that do not improve effectiveness when added to a learned model (known as delta effectiveness) are not useful, regardless of their independent effectiveness and similarity in terms of DT. Based on the dimensions of this characterisation, Figure 5 presents the five WMWD features with respect to the BM25 sample for the CW09B query set[12] and the AFS learned model. Moreover, recall from Section 4 that the sample weighting model for CW09B excludes anchor text, explaining why BM25 on the whole document is a separate feature from the sample feature. Effectiveness is measured by NDCG@20.

On analysing Figure 5, we can observe both how each weighting model feature improves the learned model, as well as how it can be characterised in terms of chorus, skimming or dark horse effect. For instance, DPH clearly brings the highest benefit to effectiveness, with PL2 and the number of matching query terms bringing very small amounts of relative benefit. Indeed, we observe that the number of matching query terms is delineated as a feature with a dark horse effect, as it is dissimilar to the sample, both in terms of DT and independent effectiveness, yet improves the learned model. On the other hand, BM25 has a clear chorus effect,

[12]Our methodology is equally applicable to the MQ2007 and MQ2008 query sets from LETOR v4.0. However, for reasons of brevity we only report results for the CW09B query set.

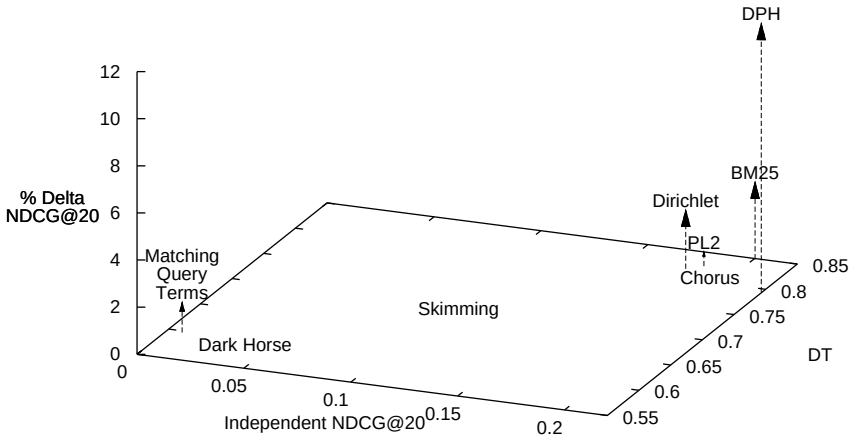

Fig. 5. Plot characterising five weighting model (WMWD) features in terms of chorus, skimming and dark horse features. Effectiveness is measured by NDCG@20.

with high effectiveness and high DT similarity to the sample. Dirichlet and DPH, as well as - to a lesser extent - PL2 also exhibit a chorus effect. Indeed, while PL2 is more independently effective than Dirichlet, and both are quite similar to the sample, it is Dirichlet that brings additional effectiveness to the learned model. For Dirichlet, this additional effectiveness is partly explained by how it handles documents with different lengths, as we will discuss in Section 6. Lastly, DPH exhibits less correlation with the sample, and is more independently effective, leading to largest improvement in effectiveness for all the WMWD features.

Overall, our observations allow us to demonstrate that features which exhibit both chorus or dark horse effects can be beneficial to the effectiveness of a learned model, although the feature with the most benefit was DPH, which was chorus in nature.

5.3 Summary

We conclude that deploying multiple weighting models as features brings new evidence that a learning to rank technique can successfully integrate into a learned model, resulting in significantly increased effectiveness. The different properties of weighting models mean that they can rank documents in different ways. Features that exhibit both chorus or dark horse effects can be used to enhance the effectiveness of a learned model: for instance, a feature with chorus characteristics (such as similar weighting models) allow a learned model to rank highly the documents that are highly scored by multiple weighting models; while features with dark horse effects (such as the number of matching query terms) may strongly indicate the likely relevance of a few documents; among the WMWD feature set, we found no features which exhibited a skimming effect.

With respect to our first research question concerning the use of multiple doc-

ument weighting models as features, the concerns of Beitzel et al. [2004] appear to be less important within a learning to rank setting. Even though the different weighting model features are all applied on the same underlying IR system implementation, document representation and query formulation, they contrast in how they rank relevant documents. In doing so, they can improve the retrieval effectiveness of a learned model. We note that as features in learning to rank re-rank the set of documents identified by the sample, the overlap aspects of data fusion identified by Lee [1997] are not present within this setting.

6. TRAINING OF DOCUMENT WEIGHTING MODEL FEATURES

In the previous section, we concluded that multiple document weighting model features add effectiveness to a learned model for Web search. Our second research question concerns the manner in which these multiple document weighting model features should be deployed, with respect to the training of their parameters. In the following, Section 6.1 experiments to address our second research question, Section 6.2 investigates the length normalisation characteristics of the learned models compared with the untrained and trained weighting model features, while Section 6.3 summarises our findings.

6.1 Experiments

Various weighting models have parameters that control the term frequency versus document length, or the smoothing. Finding appropriate settings of these parameters can significantly impact the effectiveness of the weighting models [Chowdhury et al. 2002; He and Ounis 2003; He et al. 2008]. Hence, we experiment to ascertain whether learned models that encapsulate multiple document weighting model features that have been trained are more effective than learned models with untrained document weighting model features, for which the parameters remain at their default settings as listed in Section 4.3. Indeed, there is a distinct efficiency advantage if training can be omitted without significant degradations in effectiveness.

The fat framework described in Section 4.1 provides a key advantage for answering this research question, as the postings for all the documents in the samples of the training topic sets can be easily held in memory. Hence, for repeated evaluation of different parameter settings during training, there is no need to resort to accessing the inverted file. Indeed, during training, for each trial parameter setting, the documents are ranked for the training queries and then evaluated. For instance, for the CW09B query set, using fat allows each training iteration to be completed in approximately one second on a single machine, rather than several minutes when using traditional inverted file disk access for the training queries in every iteration[13].

The results of our experiments are reported in Table VIII & IX with similar notation to Tables V & VI. We use the subscripts u and t to denote untrained and trained weighting model features, respectively. The previously explained symbols ($\ll$, $<$, $=$, $>$ and $\gg$) are used to denote the statistical significance of the difference between WMWD_u and WMWD_t. Moreover, summary Table X summarises the

[13]Experiment conducted on a quad-core Intel Xeon 2.4GHz CPU, with 8GB RAM and 400GB SATA disk containing the index.

Sample	Features QD	QI	Total	# Selected	NDCG@20	NDCG@3	ERR@20
AFS							
BM25	✗	✗	1	1	0.2056	0.1932	0.0938
BM25	$WMWD_u$	✗	6	4.0	0.2392	0.2381	0.1181
BM25	$WMWD_t$	✗	6	5.0	0.2206≪	0.2220=	0.1073<
BM25	✗	✔	24	10.0	0.2606	0.2655	0.1326
BM25	$WMWD_u$	✔	29	12.8	0.2791	0.2960	0.1453
BM25	WMB_t	✔	29	10.0	0.2716=	0.2801=	0.1398=
PL2	✗	✗	1	1	0.1762	0.1575	0.0795
PL2	$WMWD_u$	✗	6	4.2	0.2081	0.2109	0.1040
PL2	$WMWD_t$	✗	6	3.8	0.2035=	0.2025=	0.1011=
PL2	✗	✔	24	10.2	0.2418	0.2335	0.1168
PL2	$WMWD_u$	✔	29	12.4	0.2665	0.2809	0.1389
PL2	WMB_t	✔	29	8.0	0.2716=	0.2818=	0.1390=
RankSVM							
BM25	✗	✗	1	1	0.2056	0.1932	0.0938
BM25	$WMWD_u$	✗	6	-	0.2247	0.2088	0.1071
BM25	$WMWD_t$	✗	6	-	0.2219=	0.2095=	0.1037=
BM25	✗	✔	24	-	0.2771	0.2775	0.1361
BM25	$WMWD_u$	✔	29	-	0.2925	0.3193	0.1520
BM25	WMB_t	✔	29	-	0.2920=	0.3088=	0.1511=
PL2	✗	✗	1	1	0.1762	0.1575	0.0795
PL2	$WMWD_u$	✗	6	-	0.2105	0.1976	0.1006
PL2	$WMWD_t$	✗	6	-	0.2139=	0.2006=	0.1022=
PL2	✗	✔	24	-	0.2483	0.2667	0.1246
PL2	$WMWD_u$	✔	29	-	0.2821	0.3127	0.1484
PL2	WMB_t	✔	29	-	0.2825=	0.3144=	0.1488=
LambdaMART							
BM25	✗	✗	1	1	0.2056	0.1932	0.0938
BM25	$WMWD_u$	✗	6	6.0	0.2266	0.2018	0.1074
BM25	$WMWD_t$	✗	6	6.0	0.2279=	0.2178=	0.1090=
BM25	✗	✔	24	20.2	0.2620	0.2532	0.1277
BM25	$WMWD_u$	✔	29	22.6	0.2735	0.2790	0.1369
BM25	WMB_t	✔	29	23.6	0.2867=	0.2978=	0.1457=
PL2	✗	✗	1	1	0.1762	0.1575	0.0795
PL2	$WMWD_u$	✗	6	6.0	0.2110	0.2028	0.1004
PL2	$WMWD_t$	✗	6	6.0	0.2223=	0.2297>	0.1103=
PL2	✗	✔	24	20.4	0.2044	0.2166	0.1118
PL2	$WMWD_u$	✔	29	23.4	0.2844	0.3070	0.1467
PL2	WMB_t	✔	29	22.2	0.2592≪	0.2785=	0.1336<

Table VIII. Results on the CW09B query set comparing the training of (query dependent) multiple weighting models as features in a learned model. Statistical significance between $WMWD_u$ and $WMWD_t$ are denoted using the symbols ≪, <, =, > and ≫.

mean performances of each feature set for each corpus, across different samples and learners.

On analysing Table VIII (CW09B query set), we observe that training the weighting model features make very little change in the effectiveness of the learned models. In fact, in just under half of the cases (17 out of 36), the performance of the trained features ($WMWD_t$) are lower than $WMWD_u$, but only 4 of these decreases are significant. Of the 19 increases, none are significant. Comparing the number of

Query set	Features QD	QI	Total	# Selected	NDCG@10	NDCG@3	MAP@10
AFS							
MQ2007	✗	✗	1	1	0.3036	0.1674	0.1968
MQ2007	$WMWD_u$	✗	5	4.6	0.3651	0.2073	0.2519
MQ2007	$WMWD_t$	✗	5	4.4	0.3640=	0.2089=	0.2519=
MQ2007	✗	✔	9	2.8	0.3138	0.1718	0.2046
MQ2007	$WMWD_u$	✔	13	7.2	0.3641	0.2066	0.2508
MQ2007	$WMWD_t$	✔	13	4.8	0.3647=	0.2093=	0.2517=
MQ2008	✗	✗	1	1	0.4635	0.3179	0.3932
MQ2008	$WMWD_u$	✗	5	2.6	0.4977	0.3503	0.4290
MQ2008	$WMWD_t$	✗	5	2.4	0.4983=	0.3542=	0.4327=
MQ2008	✗	✔	9	2.2	0.4629	0.3118	0.3918
MQ2008	$WMWD_u$	✔	13	5.0	0.4961	0.3481	0.4276
MQ2008	$WMWD_t$	✔	13	2.6	0.4971=	0.3512=	0.4295=
RankSVM							
MQ2007	✗	✗	1	1	0.3036	0.1674	0.1968
MQ2007	$WMWD_u$	✗	5	-	0.3583	0.2020	0.2456
MQ2007	$WMWD_t$	✗	5	-	0.3557=	0.1973=	0.2423=
MQ2007	✗	✔	9	-	0.3075	0.1670	0.1998
MQ2007	$WMWD_u$	✔	13	-	0.3562	0.1992	0.2431
MQ2007	$WMWD_t$	✔	13	-	0.3563=	0.1987=	0.2422=
MQ2008	✗	✗	1	1	0.4635	0.3179	0.3932
MQ2008	$WMWD_u$	✗	5	-	0.5033	0.3600	0.4382
MQ2008	$WMWD_t$	✗	5	-	0.4964<	0.3497<	0.4318=
MQ2008	✗	✔	9	-	0.4659	0.3166	0.3956
MQ2008	$WMWD_u$	✔	13	-	0.5003	0.3543	0.4338
MQ2008	$WMWD_t$	✔	13	-	0.4971=	0.3505=	0.4320=
LambdaMART							
MQ2007	✗	✗	1	1	0.3036	0.1674	0.1968
MQ2007	$WMWD_u$	✗	5	5.0	0.3611	0.2060	0.2481
MQ2007	$WMWD_t$	✗	5	5.0	0.3566=	0.2015=	0.2438=
MQ2007	✗	✔	9	3.0	0.3305	0.1790	0.2150
MQ2007	$WMWD_u$	✔	13	12.0	0.3662	0.2076	0.2505
MQ2007	$WMWD_t$	✔	13	7.0	0.3603<	0.2045=	0.2459=
MQ2008	✗	✗	1	1	0.4635	0.3179	0.3932
MQ2008	$WMWD_u$	✗	5	5.0	0.4919	0.3457	0.4244
MQ2008	$WMWD_t$	✗	5	5.0	0.4936=	0.3468=	0.4260=
MQ2008	✗	✔	9	3.0	0.4727	0.3219	0.4005
MQ2008	$WMWD_u$	✔	13	12.0	0.4983	0.3488	0.4302
MQ2008	$WMWD_t$	✔	13	7.0	0.4973=	0.3480=	0.4297=

Table IX. Results on the LETOR query sets comparing the training of (query dependent) multiple weighting models as features in a learned model. Statistical significance between $WMWD_u$ and $WMWD_t$ are denoted using the symbols $\ll$, $<$, $=$, $>$ and $\gg$.

features selected by AFS and LambdaMART, we observe very little differences between $WMWD_u$ and $WMWD_t$: two increases, four decreases, and two learned models that selected the same number of features. Similar observations are made for the LETOR query sets in Table VIII, where only 12 out of 36 cases result in increased effectiveness with the trained features (none by a statistically significant margin), while 23 cases exhibit decreases, two by significant margins. Moreover, comparing the learned models between the $WMWD_u$ and $WMWD_t$ feature sets

Feature Set						
QI	✗	✔	✗	✔	✗	✔
CW09B	Mean NDCG@20		Mean NDCG@3		Mean ERR@20	
(sample)	0.1909	0.2687	0.1753	0.2816	0.0866	0.1375
$WMWD_u$	**0.2200** +15.2%	**0.2797** +4.1%	0.2100 +19.8%	**0.2992** +6.3%	**0.1063** +22.7%	**0.1447** +5.2%
$WMWD_t$	0.2183 +14.3%	0.2773 +3.2%	**0.2137** +21.9%	0.2936 +4.2%	0.1056 +21.9%	0.1430 +4.0%
LETOR	Mean NDCG@10		Mean NDCG@3		Mean MAP@10	
(sample)	0.3836	0.4171	0.2426	0.2664	0.2950	0.3264
$WMWD_u$	**0.4296** +12.0%	**0.4302** +3.1%	**0.2785** +14.7%	**0.2774** + 4.1%	**0.3395** +15.1%	**0.3393** +4.0%
$WMWD_t$	0.4274 +11.4%	0.4288 + 2.8%	0.2764 +13.9%	0.2770 +3.9%	0.3381 +14.6%	0.3385 +3.7%

Table X. Summary table for Table VIII & IX, detailing the mean effectiveness for different feature sets, across different samples and learners.

used by the AFS and LambdaMART learners, we observe that the number of selected features also decreases in 6 out of 8 cases.

Comparing across the learners, we observe in Table VIII that LambdaMART exhibits the most number of increases in effectiveness (9 out of 12 cases), compared to 3 and 6 cases for AFS and RankSVM, respectively - yet recall that none of these increases are significant. This is likely the result of the regression tree-based LambdaMART learner being able to create more effective learned models compared to the linear AFS or RankSVM. However, for the LETOR query sets (Table IX), the AFS learner exhibits the most number of increases in effectiveness (9, versus 1 for RankSVM and 3 for LambdaMART). Given the small margins of these improvements, we conclude that the choice of learner has no great impact on the role of training the parameters of the user weighting model features. Finally, for the CW09B query set[14], the number of increases in effectiveness observed for the different sample weighting models are widely similar (7 out of 18 for BM25, 12 for PL2), suggesting that the choice of the sample weighting model has no impact on our conclusions.

The overall performances in summary Table X are indicative of the observed trends: for both CW09B and the LETOR query sets, any training of the weighting model features generally results in overall decreased effectiveness.

6.2 Analysis

Our results above suggest that the training of the weighting model parameters, which typically control the amount of length normalisation applied, is unnecessary in a learning to rank setting. In this section, we explain why the training of these normalisation parameters is not beneficial to learned model effectiveness. Indeed, to analyse the impact of the length normalisation parameters of the weighting model features, we measure how the parameters affect any tendency to highly score shorter or longer documents, and compare this with the effectiveness of the feature. In particular, for a given test query, we measure the correlation (using Spearman's ρ) between the ranking of documents in the sample according to a particular weighting model feature, and when ranked by document length l. This correlation is measured before and after the weighting model's parameter has been trained.

[14]Recall that the sample model is fixed for the LETOR query sets.

	QI	BM25 sample		PL2 sample	
		ρ wrt l	NDCG@20	ρ wrt l	NDCG@20
Features					
$BM25_u$	-	0.01	0.1925	-0.01	0.1913
$BM25_t$	-	0.18	0.2005	0.27	0.2008
$PL2_u$	-	0.13	0.1596	0.10	0.1544
$PL2_t$	-	0.32	0.1834	0.32	0.1721
$Dirichlet_u$	-	0.30	0.1693	0.32	0.1622
$Dirichlet_t$	-	0.21	0.1774	0.22	0.1702
$Mean_u$		0.14		0.14	
$Mean_t$		0.24		0.27	
AFS					
$WMWD_u$	✗	0.29	0.2392	0.14	0.2081
$WMWD_t$	✗	0.37	0.2206	0.26	0.2035
$WMWD_u$	✔	0.36	0.2791	0.20	0.2791
$WMWD_t$	✔	0.45	0.2716	0.27	0.2716
RankSVM					
$WMWD_u$	✗	0.27	0.2247	0.16	0.2105
$WMWD_t$	✗	0.28	0.2219	0.21	0.2139
$WMWD_u$	✔	0.31	0.2925	0.25	0.2821
$WMWD_t$	✔	0.36	0.2920	0.28	0.2825
LambdaMART					
$WMWD_u$	✗	0.20	0.2266	0.09	0.2110
$WMWD_t$	✗	0.31	0.2279	0.18	0.2223
$WMWD_u$	✔	0.27	0.2735	0.20	0.2844
$WMWD_t$	✔	0.28	0.2867	0.23	0.2592
$Mean_u$		0.28		0.18	
$Mean_t$		0.34		0.24	

Table XI. Mean correlation, in terms of Spearman's ρ across all CW09B test queries between feature/learned model scores and document length, along with corresponding NDCG@20 performance.

For the CW09B query set[15], the top part of Table XI shows the mean correlation with across all test queries for three WMWD weighting model features, namely BM25, PL2 and Dirichlet – we omit the two parameter-free features, namely DPH and number of matching query terms – with parameters untrained and trained. In addition, for each feature, its untrained and trained NDCG@20 test performance is also shown. Finally, a summary mean obtained by averaging all of the untrained and trained correlations ($Mean_u$ and $Mean_t$, respectively) are also shown.

Examining the three WMWD weighting model features, we find that for BM25 and PL2, the correlation with document length increases on training of the parameter. Moreover, as expected, the test effectiveness always increases after training. Indeed, He and Ounis [2007b] found that the default $b = 0.75$ and $c = 1$ for the BM25 and PL2 weighting models are harsh normalisation settings that penalise long documents.

On the other hand, Dirichlet exhibits a higher correlation with document length than BM25 and PL2, while the correlation of Dirichlet with l actually decreases when μ is trained. Indeed, smoothing has a role in document length normalisa-

[15]This methodology can equally be applied to the LETOR v4.0 query sets, which are omitted for reasons of brevity.

tion [Smucker and Allan 2005], but our results suggest that the default parameter setting for Dirichlet has a preference for highly scoring longer documents. Overall, from the average of the untrained and trained correlations (Mean_u and Mean_t, respectively), we can see that training the weighting model parameters increases the correlations to $\rho = 0.24..0.27$ on average.

Next, we contrast the correlations of the weighting model features with any those exhibited by the learned models. Both correlations with document length and performances for each learned model are shown in the lower parts of Table XI. From the table, we note that the correlation for all learned models are typically higher than most of the untrained features alone. Indeed, even without training, the correlations exhibited by the learned models for the BM25 sample can rise as high as 0.29. However, the correlations of the learned models using the trained weighting model features are higher still, despite exhibiting no marked increases in effectiveness. This suggests that the learned models using the trained features may not be penalising long documents sufficiently. Indeed, a high correlation with l suggests a bias towards longer documents, and means that the relevance scores of long documents have been over-estimated.

The general trend from Table XI demonstrates that the learners intrinsically account for document length bias during learning. This is because the untrained weighting model features exhibit different length normalisation properties, and hence an over-estimated relevance score by one feature can be counteracted by stronger application of another feature that under-estimates the relevance score with respect to document length.

This effect can be explained for the various learners as follows: For RankSVM, as the training examples are pairs of documents, every over-estimated relevance score is subtracted by another over-estimated relevance score. For LambdaMART, the scenario is similar. In particular, LambdaMART is an improved version of LambdaRank, which is a listwise algorithm based on the pairwise RankNet [Burges et al. 2005]. A subtraction of two documents' relevance scores is defined in its loss function such that it is only the relative difference between scores matters. For a listwise technique such as AFS, the inflation of relevance scores can be controlled by the linear combination weights for different features. Indeed, an over-estimated relevance score by a weighting model may be addressed by the learner through a higher than expected feature weight for another weighting model that under-estimates the relevance score with respect to document length.

Overall, we conclude that any document length bias present in the weighting model features is implicitly handled by the learners, by the nature of their combination of multiple weighting model features, due to the diverse length normalisation effects exhibited by different weighting models. On the other hand, by training the weighting model features prior to learning, the ability of the learners to combine multiple weighting models to address document length bias is hindered, as the weighting models features have more similar length normalisation effects. This results in learned models that may be overfitted to the training queries.

6.3 Summary

With respect to our second research question concerning the training of the parameters of document weighting model features within a learned model, our observations

from Section 6.1 above find that this is unnecessary for effective learned models. Indeed, from Section 6.2 we conclude that training the parameters of weighting models results in changes to the bias in document length that the learning to rank technique would have anyway accounted for, and hence does not lead to significantly improved learned models using those trained features. Hence, for effective learned models, it is not necessary to train the weighting model features. As a result, deployments of learned models can avoid such costly training operations.

7. FEATURES BASED ON WEIGHTING MODELS FOR FIELDS

Our final research question is concerned with the treatment of fields within a learned model. In particular, Robertson et al. [2004] contrasted linearly combining model scores for each field with field-based weighting models – where the (possibly normalised [Zaragoza et al. 2004; Macdonald et al. 2006]) term frequency in each field is linearly combined before scoring. They argued, with theoretical motivations (summarised in Section 3.3 above), as well as with empirical evidence, that field-based models should be used in preference to linearly combining weighting models for each field. However, this work pre-dated the introduction of learning to rank and contrasts with recent learning to rank deployments where weighting models are typically calculated on each field in isolation (e.g. LETOR v3.0 [Qin et al. 2009] and v4.0), for later combination within the learned model. In the following, we experiment to investigate the most effective approach for combining fields within a learned model (Section 7.1), characterise the various field features as introducing chorus, skimming or dark horse effects (Section 7.2) and summarise our findings (Section 7.3).

7.1 Experiments

To address our final research question, we measure the difference in the effectiveness of the learned models that have one weighting model for each field (Equation (4)), versus a field-based weighting model (Equation (7)), for which we use feature sets WMSF and WMFB respectively from Section 4.2. In both cases, all four fields illustrated by Figure 2 are used: title, URL, body and anchor text. Moreover, recall from Section 4.2 that we use BM25 or PL2 as the models for the single-field features, and the related BM25F or PL2F as the field-based weighting model features. As neither BM25F nor PL2F have stated default parameter settings, we resort to training the length normalisation parameters for each field (b_f for BM25F in Equation (10), or c_f for PL2F in Equation (13)) and the weight of each field (w_f). Finally, to ensure a fair setting, the length normalisation parameters for each of the single-field model features are also trained.

Results are reported in Tables XII & XIII, grouped by the four different choices for query dependent features (i.e. none, WMSF, WMFB and WMSF+WMFB). Significance is measured according to the four rows within each group, so that each row contains symbol(s) denoting the significance compared to the rows above it. Once again, for each corpus, the mean performance of each feature set averaged across different samples, weighting models and learners is reported in summary Table XIV.

From Table XII, we observe that for the CW09B query set, adding weighting models computed on each field as features can benefit the effectiveness of the learned models (only 5 decreases in performance across 36 cases), with significant increases

Sample	Features QD	QI	Total	# Selected	NDCG@20	NDCG@3	ERR@20
AFS							
PL2	✗	✗	1	1	0.1762	0.1575	0.0795
PL2	WMSF(PL2)$_t$	✗	5	3.4	0.1892=	0.1632=	0.0818=
PL2	WMFB(PL2F)$_t$	✗	2	2.0	0.2075≫=	0.1732==	0.0922>=
PL2	WMSF(PL2)$_t$ + WMFB(PL2F)$_t$	✗	6	3.8	0.2147≫==	0.1897===	0.0986≫>=
PL2	✗	✔	24	10.2	0.2418	0.2335	0.1168
PL2	WMSF(PL2)$_t$	✔	40	11.0	0.2379=	0.2657>	0.1224=
PL2	WMFB(PL2F)$_t$	✔	25	7.2	0.2786>≫	0.2855>=	0.1433≫>
PL2	WMSF(PL2)$_t$ + WMFB(PL2F)$_t$	✔	29	7.2	0.2764>≫=	0.2811>==	0.1417≫>=
BM25	✗	✗	1	1	0.2056	0.1932	0.0938
BM25	WMSF(BM25)$_t$	✗	5	3.8	0.2213=	0.2156=	0.1073>
BM25	WMFB(BM25F)$_t$	✗	2	1.4	0.2197==	0.1812==	0.0989==
BM25	WMSF(BM25)$_t$ + WMFB(BM25F)$_t$	✗	6	4.2	0.2272>==	0.2017===	0.1044===
BM25	✗	✔	24	10.0	0.2606	0.2655	0.1326
BM25	WMSF(BM25)$_t$	✔	28	10.8	0.2809=	0.2999=	0.1444=
BM25	WMFB(BM25F)$_t$	✔	25	11.6	0.2612==	0.2657=<	0.1316=<
BM25	WMSF(BM25)$_t$ + WMFB(BM25F)$_t$	✔	29	13.0	0.2645===	0.2730===	0.1367===
RankSVM							
PL2	✗	✗	1	1	0.1762	0.1575	0.0795
PL2	WMSF(PL2)$_t$	✗	5	-	0.1884=	0.1736=	0.0826=
PL2	WMFB(PL2F)$_t$	✗	2	-	0.2057≫=	0.1753==	0.0914>=
PL2	WMSF(PL2)$_t$ + WMFB(PL2F)$_t$	✗	6	-	0.2209≫>≫	0.1920===	0.0996≫>≫
PL2	✗	✔	24	-	0.2483	0.2667	0.1246
PL2	WMSF(PL2)$_t$	✔	40	-	0.2422=	0.2659=	0.1257=
PL2	WMFB(PL2F)$_t$	✔	25	-	0.2911≫≫	0.3064>>	0.1451≫≫
PL2	WMSF(PL2)$_t$ + WMFB(PL2F)$_t$	✔	29	-	0.2824≫≫=	0.3105>>=	0.1456≫≫=
BM25	✗	✗	1	1	0.2056	0.1932	0.0938
BM25	WMSF(BM25)$_t$	✗	5	-	0.2143=	0.2119=	0.1033=
BM25	WMFB(BM25F)$_t$	✗	2	-	0.2198==	0.1869==	0.0992==
BM25	WMSF(BM25)$_t$ + WMFB(BM25F)$_t$	✗	6	-	0.2268>≫=	0.2065===	0.1062===
BM25	✗	✔	24	-	0.2771	0.2775	0.1361
BM25	WMSF(BM25)$_t$	✔	28	-	0.2815=	0.3007=	0.1437=
BM25	WMFB(BM25F)$_t$	✔	25	-	0.2789==	0.2974>=	0.1431>=
BM25	WMSF(BM25)$_t$ + WMFB(BM25F)$_t$	✔	29	-	0.2788===	0.3066===	0.1450===
LambdaMART							
PL2	✗	✗	1	1	0.1762	0.1575	0.0795
PL2	WMSF(PL2)$_t$	✗	5	5.0	0.1802=	0.1674=	0.0854=
PL2	WMFB(PL2F)$_t$	✗	2	2.0	0.1973==	0.1637==	0.0878==
PL2	WMSF(PL2)$_t$ + WMFB(PL2F)$_t$	✗	6	6.0	0.2138≫≫=	0.2037===	0.1023>>>
PL2	✗	✔	24	20.4	0.2044	0.2166	0.1118
PL2	WMSF(PL2)$_t$	✔	40	23.2	0.2299≫	0.2266=	0.1174=
PL2	WMFB(PL2F)$_t$	✔	25	21.6	0.2748≫≫	0.2845≫≫	0.1393≫≫
PL2	WMSF(PL2)$_t$ + WMFB(PL2F)$_t$	✔	29	24.8	0.2733≫≫=	0.2817≫≫=	0.1344≫≫=
BM25	✗	✗	1	1	0.2056	0.1932	0.0938
BM25	WMSF(BM25)$_t$	✗	5	5.0	0.2024=	0.1800=	0.0973=
BM25	WMFB(BM25F)$_t$	✗	2	2.0	0.2130==	0.2044==	0.1030==
BM25	WMSF(BM25)$_t$ + WMFB(BM25F)$_t$	✗	6	6.0	0.2116===	0.2005===	0.1012===
BM25	✗	✔	24	20.2	0.2620	0.2532	0.1277
BM25	WMSF(BM25)$_t$	✔	28	24.2	0.2826=	0.3094≫	0.1477>
BM25	WMFB(BM25F)$_t$	✔	25	20.2	0.2762==	0.3055≫=	0.1447>=
BM25	WMSF(BM25)$_t$ + WMFB(BM25F)$_t$	✔	29	24.8	0.2785===	0.3037>==	0.1474≫==

Table XII. Results on the CW09B query set comparing the use of weighting models computed on each single-field versus a field-based weighting model as features in a learned model. On each row, denoted using the usual symbols, there are k significance tests compared to the k rows above. For instance, for WMFB(BM25F)$_t$, significant differences are shown compared to the 1 feature sample and compared to WMSF(BM25)$_t$, respectively.

observed in three cases. Using a field-based weighting model (i.e. PL2F or BM25F) as a feature instead of the weighting models computed on each field almost always results in improved effectiveness (significantly improved in 17 cases). Indeed, PL2F or BM25F can even improve on the single-field models in 18 out of 36 cases, 8 significantly so.

However, analysing the results for the LETOR query sets in Table XIII produces some contrasting results. In particular, adding single-field or field-based weighting

Query set	Features QD	QI	Total	# Selected	NDCG@10	NDCG@3	MAP@10
AFS							
MQ2007	✗	✗	1	1	0.3036	0.1674	0.1968
MQ2007	WMSF(BM25)$_t$	✗	5	3.6	0.3583≫	0.2056≫	0.2408≫
MQ2007	WMFB(BM25F)$_t$	✗	2	2.0	0.3489≫≪	0.1932≫≪	0.2339≫<
MQ2007	WMSF(BM25)$_t$ + WMFB(BM25F)$_t$	✗	6	4.8	0.3594≫=≫	0.2058≫=≫	0.2420≫=≫
MQ2007	✗	✔	3	2.8	0.3138	0.1718	0.2046
MQ2007	WMSF(BM25)$_t$	✔	7	4.8	0.3578≫	0.2039≫	0.2400≫
MQ2007	WMFB(BM25F)$_t$	✔	3	2.4	0.3467≫≪	0.1896≫≪	0.2331≫=
MQ2007	WMSF(BM25)$_t$ + WMFB(BM25F)$_t$	✔	8	5.4	0.3588≫=≫	0.2043≫=≫	0.2411≫=>
MQ2008	✗	✗	1	1	0.4635	0.3179	0.3932
MQ2008	WMSF(BM25)$_t$	✗	5	2.8	0.4970≫	0.3557≫	0.4302≫
MQ2008	WMFB(BM25F)$_t$	✗	2	2.0	0.4843≫≪	0.3376≫≪	0.4152≫≪
MQ2008	WMSF(BM25)$_t$ + WMFB(BM25F)$_t$	✗	6	3.6	0.4973≫=≫	0.3545≫=≫	0.4314≫=≫
MQ2008	✗	✔	3	2.2	0.4629	0.3118	0.3918
MQ2008	WMSF(BM25)$_t$	✔	7	4.2	0.4978≫	0.3542≫	0.4308≫
MQ2008	WMFB(BM25F)$_t$	✔	3	2.2	0.4891≫<	0.3471≫=	0.4246≫=
MQ2008	WMSF(BM25)$_t$ + WMFB(BM25F)$_t$	✔	8	3.0	0.4960≫==	0.3551≫==	0.4302≫==
RankSVM							
MQ2007	✗	✗	1	1	0.3036	0.1674	0.1968
MQ2007	WMSF(BM25)$_t$	✗	5	-	0.3607≫	0.2049≫	0.2429≫
MQ2007	WMFB(BM25F)$_t$	✗	2	-	0.3489≫≪	0.1938≫≪	0.2340≫≪
MQ2007	WMSF(BM25)$_t$ + WMFB(BM25F)$_t$	✗	6	-	0.3615≫=≫	0.2057≫=≫	0.2439≫=≫
MQ2007	✗	✔	3	-	0.3075	0.1670	0.1998
MQ2007	WMSF(BM25)$_t$	✔	7	-	0.3604≫	0.2045≫	0.2411≫
MQ2007	WMFB(BM25F)$_t$	✔	3	-	0.3399≫≪	0.1815≫≪	0.2275≫≪
MQ2007	WMSF(BM25)$_t$ + WMFB(BM25F)$_t$	✔	8	-	0.3621≫=≫	0.2063≫=≫	0.2445≫≫≫
MQ2008	✗	✗	1	1	0.4635	0.3179	0.3932
MQ2008	WMSF(BM25)$_t$	✗	5	-	0.4973≫	0.3505≫	0.4290≫
MQ2008	WMFB(BM25F)$_t$	✗	2	-	0.4840≫≪	0.3366≫=	0.4158≫<
MQ2008	WMSF(BM25)$_t$ + WMFB(BM25F)$_t$	✗	6	-	0.4965≫=≫	0.3506≫==	0.4290≫=>
MQ2008	✗	✔	3	-	0.4659	0.3166	0.3956
MQ2008	WMSF(BM25)$_t$	✔	7	-	0.5052≫	0.3633≫	0.4400≫
MQ2008	WMFB(BM25F)$_t$	✔	3	-	0.4875≫≪	0.3454≫<	0.4230≫≪
MQ2008	WMSF(BM25)$_t$ + WMFB(BM25F)$_t$	✔	8	-	0.5009≫<≫	0.3563≫<=	0.4341≫≪=
LambdaMART							
MQ2007	✗	✗	1	1	0.3036	0.1674	0.1968
MQ2007	WMSF(BM25)$_t$	✗	5	4.8	0.3570≫	0.2041≫	0.2392≫
MQ2007	WMFB(BM25F)$_t$	✗	2	2.0	0.3464≫≪	0.1923≫≪	0.2311≫≪
MQ2007	WMSF(BM25)$_t$ + WMFB(BM25F)$_t$	✗	6	6.0	0.3586≫=≫	0.2054≫=≫	0.2409≫=≫
MQ2007	✗	✔	3	3.0	0.3305	0.1790	0.2150
MQ2007	WMSF(BM25)$_t$	✔	7	7.0	0.3598≫	0.2029≫	0.2405≫
MQ2007	WMFB(BM25F)$_t$	✔	3	3.0	0.3524≫<	0.1966≫=	0.2364≫=
MQ2007	WMSF(BM25)$_t$ + WMFB(BM25F)$_t$	✔	8	8.0	0.3653≫>≫	0.2084≫>≫	0.2468≫≫≫
MQ2008	✗	✗	1	1	0.4635	0.3179	0.3932
MQ2008	WMSF(BM25)$_t$	✗	5	5.0	0.4951≫	0.3492≫	0.4268≫
MQ2008	WMFB(BM25F)$_t$	✗	2	2.0	0.4810≫≪	0.3329>≪	0.4132≫≪
MQ2008	WMSF(BM25)$_t$ + WMFB(BM25F)$_t$	✗	6	6.0	0.4968≫=≫	0.3546≫=≫	0.4295≫=≫
MQ2008	✗	✔	3	3.0	0.4727	0.3219	0.4005
MQ2008	WMSF(BM25)$_t$	✔	7	6.8	0.4916≫	0.3497≫	0.4230≫
MQ2008	WMFB(BM25F)$_t$	✔	3	3.0	0.4865≫=	0.3429≫=	0.4178≫=
MQ2008	WMSF(BM25)$_t$ + WMFB(BM25F)$_t$	✔	8	8.0	0.4961≫=>	0.3506≫==	0.4282≫=>

Table XIII. Results on the LETOR query sets comparing the use of weighting models computed on each single-field versus a field-based weighting model as features in a learned model. On each row, denoted using the usual symbols, there are k significance tests compared to the k rows above. For instance, for WMFB(BM25F)$_t$, significant differences are shown compared to the 1 feature sample and compared to WMSF(BM25)$_t$, respectively.

models as features always results in a significant improvement over the sample. However, in contrast to Table XII, the field-based models do not outperform single-field models, and are significantly worse in 27 out of 36 cases. These observations can also be made from summary Table XIV, which shows smaller margins of improvement over the sample for WMFB than for WMSF for the LETOR query sets, and the opposite for the CW09B query sets. These results suggest that the field-based weighting models do not provide as useful evidence on the LETOR query sets.

Feature Set						
QI	✗	✔	✗	✔	✗	✔
CW09B	Mean NDCG@20		Mean NDCG@3		Mean ERR@20	
(sample)	0.1909	0.2652	0.1753	0.2785	0.0866	0.1354
$WMSF_t$	0.1993 +4.4%	0.2592 -2.3%	0.1853 +5.7%	0.2780 -0.2%	0.0930 +7.3%	0.1335 -1.4%
$WMFB_t$	0.2148 +12.5%	**0.2762** +4.1%	0.1899 +8.3%	0.2918 +4.8%	0.0987 +14.0%	0.1415 +4.5%
$WMFB_t$+$WMSF_t$	**0.2192** +14.8%	0.2757 +3.9%	**0.1990** +13.5%	**0.2928** +5.1%	**0.1020** +17.7%	**0.1418** +4.7%
LETOR	Mean NDCG@10		Mean NDCG@3		Mean MAP@10	
(sample)	0.3836	0.4170	0.2426	0.2679	0.2950	0.3254
$WMSF_t$	0.4276 +11.5%	0.4288 +2.8%	0.2783 +14.7%	0.2797 +4.4%	0.3348	0.3359 +3.2%
$WMFB_t$	0.4220 10.0%	0.4234 +1.5%	0.2719 +12.1%	0.2737 +2.2%	0.3300	0.3323 +2.1%
$WMFB_t$ + $WMSF_t$	**0.4283** +11.6%	**0.4299** +3.1%	**0.2794** +15.2%	**0.2802** +4.5%	**0.3361** +13.9%	**0.3375** +3.7%

Table XIV. Summary table for Tables XII & XIII, detailing the mean performances on each corpus for different feature sets, across different samples and learners.

Next, when weighting models computed on each field and the field-based models are all deployed as features, performance can be further increased. On the CW09B query set, the increases when WMFB+WMSF features are added to learned models that have QI features are less marked (see Table XIV). This suggests that the benefit brought by the weighting models computed on each field are encapsulated by the query independent features. To illustrate this, consider a document that has an amount of anchor text related to the query, as identified by a single model on the anchor text field. Our results suggest that it may be sufficient for the learned model to know that the document has a significant amount of anchor text (as measured by QI features such as field length, or inlink count), and that the document is about the query (c.f. PL2F or BM25F). On the other hand, we observe no discernible pattern concerning the margin of effectiveness improvement brought by WMFB+WMSF with/without QI features for the LETOR query sets.

Moreover, contrasting across the different learners in Tables XII & XIII, we observe no marked difference in the effectiveness between the use of linear learned models (as obtained from AFS or RankSVM) compared to LambdaMART, which is based on regression trees. This shows that our empirical findings are robust to the type of learner deployed, and that the type of learned model (linear combination or regression tree) does not impact on the earlier arguments of Robertson et al. [2004].

Overall, we conclude that for creating effective learned models for Web search tasks, the use of a field-based weighting model can be more effective than computing weighting model features on each field. These observations are easily observable in summary Table XIV for the CW09B query set, with $WMFB_t$ models performing markedly higher than $WMSF_t$, and $WMFB_t$ + $WMSF_t$ showing only a little added value above $WMFB_t$. However, the benefit for the LETOR query sets is less marked - we postulate that anchor text is less useful for the GOV2 collection and/or the Million Query track query sets. This could be because the queries are informational in nature, where anchor text is less important than for navigational queries [Croft et al. 2009]. However, we also note that the system used to create the LETOR v4.0 has not identified much anchor text within the GOV2 corpus - indeed, comparing the statistics in Table III with those reported in [He and Ounis 2007a] shows that LETOR v4.0 anchor text token estimates differ by an order of magnitude (86M vs. 966M tokens, equating to average anchor text field length of 3.5 vs. 32.3 tokens). With such a low level of anchor text, the benefits brought by per-field normalisation are unlikely to be needed, explaining why the single-field features are more effective.

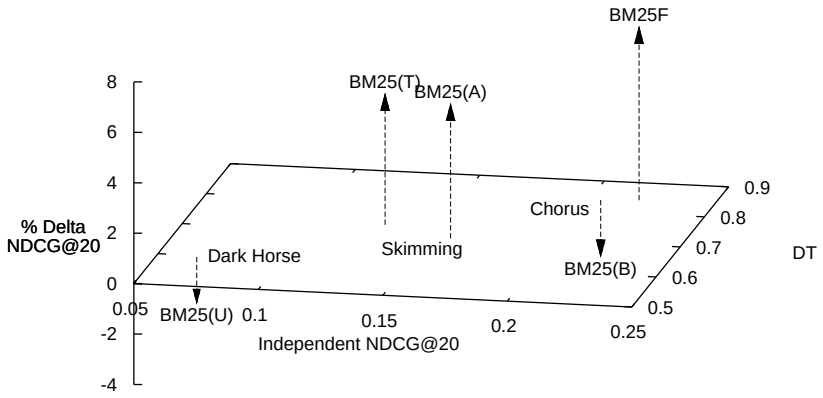

Fig. 6. Plot characterising five WMSF and WMFB features in terms of chorus, skimming and dark horse features. Effectiveness is measured by NDCG@20. BM25 WMSF feature are parametrised by their field (Title, URL, Body or Anchor Text).

The results in this section consolidate the theoretical argument of Robertson et al. [2004] (see Section 3.3) showing the appropriateness of field-based weighting models, even within a learning to rank setting. Indeed, we conclude that field-based weighting models should be deployed for effective retrieval. A related consequence is that the existing learning to rank datasets (LETOR and MSLR-WEB10K, etc) could be improved by the introduction of field-based weighting models such as BM25F and PL2F.

7.2 Analysis

To understand the impact of the various field weighting model features on effectiveness of the learned model, and why they have such impact, we again characterise the features in terms of dark horse, skimming or chorus effects, as proposed in Section 5.2. Figure 6 shows the independent effectiveness, DT and delta effectiveness for the AFS learner and the BM25 sample on the CW09B query set of BM25F and the the four BM25 single-field features (denoted BM25(T) for title, BM25(A) for anchor text, BM25(B) for body and BM25(U) for URL)[16].

On analysing Figure 6, we observe that calculating BM25 on the URL field exhibits a dark horse effect, as it does not positively impact effectiveness. On the other hand, calculating BM25 on the title and anchor text fields exhibit skimming effects - they retrieve different relevant documents to the sample, but dramatically improve a learned model. We note that this confirms the early work of Katzer et al. [1982] on different document representations. BM25 on the body alone appears

[16]We omit the figures for the LETOR query sets for reasons of brevity.

to lead to a chorus effect but does not help to improve effectiveness. Finally, BM25F exhibits a strong chorus effect - this is likely as it integrates the evidence brought by BM25(T) and BM25(A) at lower ranks. Indeed, a feature with high effectiveness like BM25F has a chorus-like effect, even if it integrates other features with skimming properties. Further empirical validation, e.g. through empirical ablation studies could better determine the dependencies between features with chorus and skimming properties.

7.3 Summary

For our third research question, we examined how weighting model scores obtained from different document representations should be combined. Our results show that field-based weighting models, such as PL2F or BM25F, can be more appropriate than their corresponding single-field weighting models on the CW09B query set. This consolidates the earlier arguments of Robertson et al. [2004] concerning the advantages of field-based models to a learning to rank setting, where they are shown to hold regardless of the applied learning to rank technique. Indeed, we find that a field-based model exhibits an chorus effect, providing effective reinforcing evidence for the learner. However, for the LETOR query sets, field-based weighting models tend to be significantly less effective. This suggests that the per-field normalisation brought by such weighting models is not needed, probably due to the sparseness of the anchor text in LETOR v4.0, as discussed above.

8. CONCLUSIONS

This paper concerns the deployment of effective ranking models obtained using learning to rank, particularly when these contain multiple weighting models as query dependent features. We investigated three particular questions within this context, namely the deployment of multiple weighting model features, the setting of their parameters, and, finally, the proper use of weighting models considering different document representations or fields. By doing so, we are able to make empirically justified recommendations of the best practices for deploying multiple query dependent weighting models within learning to rank deployments.

Indeed, while some existing learning to rank datasets appear to use multiple document weighting model features, there exists no empirical validation of the effectiveness of such a setting in the literature. Our results in Section 5 attest the effectiveness of deploying multiple document weighting models as features - contrasting with earlier studies within the data fusion literature. Moreover, we show how different weighting model features can be characterised as exhibiting chorus, skimming or dark horse effects within the learned model.

In Section 6, we show that the training of the parameters of these multiple document weighting model features is unnecessary and hence can be safely omitted when building a search engine in a learning to rank setting. This is explained in that any correction of document length penalisation obtained by training of a weighting model's parameters is implicitly carried out by the learning to rank technique during the learning process.

Finally, while the LETOR learning to rank datasets deploy weighting models calculated on each field individually as features, we showed in Section 7 that, for

the ClueWeb09 corpus and corresponding query set, such features are inferior to deploying a field-based weighting model such as PL2F or BM25F on their own. This consolidates the arguments of Robertson et al. [2004] which pre-dates the introduction of learning to rank. However, for the LETOR v4.0 query sets, the field-based weighting models were not effective - this may be due to the sparsity of anchor text within LETOR v4.0 that is not observed by another work on the underlying GOV2 collection [He and Ounis 2007a].

To facilitate the experiments in this paper, we introduced the fat framework that we use, which permits the calculation of multiple weighting model features without resorting to multiple accesses to the inverted file posting lists on disk. In particular, the fat framework "fattens" result sets with the postings of sample documents, such that additional features can be calculated using these postings at a later stage in the retrieval process. This framework can be used both for efficient and effective search engine deployments, as well as to increase the efficiency of experimentation within a learning to rank context.

REFERENCES

Amati, G. 2003. Probabilistic models for information retrieval based on divergence from randomness. Ph.D. thesis, Department of Computing Science, University of Glasgow.

Amati, G., Ambrosi, E., Bianchi, M., Gaibisso, C., and Gambosi, G. 2008. FUB, IASI-CNR and University of Tor Vergata at TREC 2007 Blog track. In *TREC '07: Proceedings of the 16th Text REtrieval Conference*, Gaithersburg, MD, USA.

Becchetti, L., Castillo, C., Donato, D., Leonardi, S., and Baeza-Yates, R. 2006. Link-based characterization and detection of Web spam. In *Proc. of AIRWeb*.

Beitzel, S. M., Jensen, E. C., Chowdhury, A., Grossman, D., Frieder, O., and Goharian, N. 2004. Fusion of effective retrieval strategies in the same information retrieval system. *J. Am. Soc. Inf. Sci. Technol. 55,* 10, 859–868.

Bendersky, M., Croft, W. B., and Diao, Y. 2011. Quality-biased ranking of web documents. In *WSDM '11: Proceedings of the fourth ACM international conference on Web search and data mining*. ACM, New York, NY, USA, 95–104.

Broder, A. Z., Carmel, D., Herscovici, M., Soffer, A., and Zien, J. 2003. Efficient query evaluation using a two-level retrieval process. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*. ACM Press, New York, NY, USA, 426–434.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. 2005. Learning to rank using gradient descent. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*. ACM, New York, NY, USA, 89–96.

Callan, J., Hoy, M., Yoo, C., and Zhao, L. 2009. The ClueWeb09 dataset. In *TREC '09: Proceedings of the 18th Text REtrieval Conference*. Gaithersburg, MD, USA.

Cambazoglu, B. B., Zaragoza, H., Chapelle, O., Chen, J., Liao, C., Zheng, Z., and Degenhardt, J. 2010. Early exit optimizations for additive machine learned ranking systems. In *WSDM '10: Proceedings of the third ACM international conference on Web search and data mining*. WSDM '10. ACM, New York, NY, USA, 411–420.

Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., and Li, H. 2007. Learning to rank: from pairwise approach to listwise approach. In *ICML '07: Proceedings of the 24th international conference on Machine learning*. ACM, New York, NY, USA, 129–136.

Chappelle, O. and Chang, Y. 2011. Yahoo! learning to rank challenge overview. In *JMLR Workshop and Conference Proceedings*. Number 14. 1–24.

Chowdhury, A., McCabe, M. C., Grossman, D., and Frieder, O. 2002. Document normalization revisited. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, New York, NY, USA, 381–382.

Clarke, C. L. A., Craswell, N., and Soboroff, I. 2010. Overview of the TREC 2009 Web track. In *TREC '09: Proceedings of the 18th Text REtrieval Conference (TREC 2009)*. Gaithersburg, MD, USA.

Clarke, C. L. A., Craswell, N., and Soboroff, I. 2011. Overview of the TREC 2010 Web track. In *TREC '10: Proceedings of the 19th Text REtrieval Conference*. Gaithersburg, MD, USA.

Clarke, C. L. A., Craswell, N., and Soboroff, I. 2012. Overview of the TREC 2011 Web track. In *TREC '11: Proceedings of the 20th Text REtrieval Conference (TREC 2011)*. Gaithersburg, MD, USA.

Cormack, G. V., Smucker, M. D., and Clarke, C. L. A. 2011. Efficient and effective spam filtering and re-ranking for large Web datasets. *Inf. Retr. 14,* 5, 441–465.

Craswell, N., Fetterly, D., Najork, M., Robertson, S., and Yilmaz, E. 2010. Microsoft Research at TREC 2009. In *TREC '09: Proceedings of the 18th Text REtrieval Conference (TREC 2009)*. Gaithersburg, MD, USA.

Croft, B., Metzler, D., and Strohman, T. 2009. *Search Engines: Information Retrieval in Practice*. Addison-Wesley Publishing Company, USA.

Fang, H., Tao, T., and Zhai, C. 2004. A formal study of information retrieval heuristics. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, New York, NY, USA, 49–56.

Fox, E. A., Koushik, P., Shaw, J. A., Modlin, R., and Rao, D. 1993. Combining evidence from multiple searches. In *TREC-1: Proceedings of the First Text REtrieval Conference*. 319–328. NIST Special Publication 500-207. Gaithersburg, MD, USA.

Fox, E. A. and Shaw, J. A. 1994. Combination of multiple searches. In *TREC-2: Proceedings of the 2nd Text REtrieval Conference*. 243–252. NIST Special Publication 500-215. Gaithersburg, MD, USA.

Ganjisaffar, Y., Caruana, R., and Lopes, C. 2011. Bagging gradient-boosted trees for high precision, low variance ranking models. In *SIGIR '11: Proceedings of the 34th international ACM SIGIR conference on Research and development in Information*. ACM, New York, NY, USA, 85–94.

He, B. 2007. Term frequency normalisation for information retrieval. Ph.D. thesis, Department of Computing Science, University of Glasgow.

He, B., Macdonald, C., and Ounis, I. 2008. Retrieval sensitivity under training using different measures. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval* ACM, New York, NY, USA, 67–74.

He, B. and Ounis, I. 2003. A study of parameter tuning for term frequency normalization. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*. ACM Press, New York, NY, USA, 10–16.

He, B. and Ounis, I. 2005. A study of the Dirichlet Priors for term frequency normalisation. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, New York, NY, USA, 465–471.

He, B. and Ounis, I. 2007a. Combining fields for query expansion and adaptive query expansion. *Inf. Process. Manage. 43,* 5, 1294–1307.

He, B. and Ounis, I. 2007b. On setting the hyper-parameters of term frequency normalization for information retrieval. *ACM Trans. Inf. Syst. 25,* 3.

Joachims, T. 2002. Optimizing search engines using clickthrough data. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, New York, NY, USA, 133–142.

Jones, K. S., Walker, S., and Robertson, S. E. 2000. A probabilistic model of information retrieval: development and comparative experiments. *Inf. Process. Manage. 36*, 779–808.

Kamps, J. 2006. Effective smoothing for a terabyte of text. In *TREC '05: Proceedings of the 14th Text REtrieval Conference*. NIST Special Publication, Gaithersburg, MD, USA.

Katzer, J., M.J.McGill, Tessier, J., Frakes, W., and Dasgupta, P. 1982. A study of the overlap among document representations. *Information Technology: Research and Development 1,* 2, 261–274.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. 1983. Optimization by simulated annealing. *Science 220,* 4598, 671–680.

Kraaij, W., Westerveld, T., and Hiemstra, D. 2002. The importance of prior probabilities for entry page search. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, New York, NY, USA, 27–34.

Lee, J. H. 1997. Analyses of multiple evidence combination. In *SIGIR '97: Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, New York, NY, USA, 267–276.

Lempel, R. and Moran, S. 2001. SALSA: the stochastic approach for link-structure analysis. *ACM Trans. Inf. Syst. 19*, 131–160.

Li, H. 2011. *Learning to Rank for Information Retrieval and Natural Language Processing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Liu, T.-Y. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval 3,* 3, 225–331.

Macdonald, C., McCreadie, R., Santos, R., and Ounis, I. 2012. From puppy to maturity: Experiences in developing terrier. In *Proceedings of the SIGIR 2012 Workshop in Open Source Information Retrieval*.

Macdonald, C., Plachouras, V., He, B., Lioma, C., and Ounis, I. 2006. University of Glasgow at WebCLEF 2005: Experiments in per-field normlisation and language specific stemming. In *Proceedings of CLEF Workshop 2005*. Lecture Notes in Computing Science, 898–907.

Macdonald, C., Santos, R., and Ounis, I. 2012. The whens and hows of learning to rank for web search. *Information Retrieval*, 1–45. 10.1007/s10791-012-9209-9.

Metzler, D. 2007a. Using gradient descent to optimize language modeling smoothing parameters. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, New York, NY, USA, 687–688.

Metzler, D. A. 2007b. Automatic feature selection in the markov random field model for information retrieval. In *CIKM '07: Proceedings of the 16th ACM conference on Conference on information and knowledge management*. ACM, New York, NY, USA, 253–262.

Moffat, A. and Zobel, J. 1996. Self-indexing inverted files for fast text retrieval. *ACM Trans. Inf. Syst. 14,* 4, 349–379.

Ogilvie, P. and Callan, J. 2003. Combining document representations for known-item search. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. ACM, New York, NY, USA, 143–150.

Page, L., Brin, S., Motwani, R., and Winograd, T. 1998. The PageRank citation ranking: Bringing order to the web. Tech. rep., Stanford Digital Library Technologies Project.

Peng, J., Macdonald, C., He, B., Plachouras, V., and Ounis, I. 2007. Incorporating term dependency in the DFR framework. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM Press, New York, NY, USA.

Plachouras, V. 2006. Selective web information retrieval. Ph.D. thesis, Department of Computing Science, University of Glasgow.

Plachouras, V., Ounis, I., and Amati, G. 2005. The Static Absorbing Model for the Web. *Journal of Web Engineering 4,* 2, 165–186.

Qin, T., Liu, T.-Y., Xu, J., and Li, H. 2009. LETOR: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval 13,* 4, 347–374.

Robertson, S., Zaragoza, H., and Taylor, M. 2004. Simple BM25 extension to multiple weighted fields. In *CIKM '04: Proceedings of the thirteenth ACM conference on Information and knowledge management*. ACM, New York, NY, USA, 42–49.

Robertson, S. E. and Jones, K. S. 1976. Relevance weighting of search terms. *Relevance weighting of search terms 1,* 27, 129–146.

Robertson, S. E., Walker, S., Hancock-Beaulieu, M., Gatford, M., and Payne, A. 1995. Okapi at TREC-4. In *Proceedings of the 4th Text REtrieval Conference (TREC-4)*. Gaithersburg, MD, USA.

Robertson, S. E., Walker, S., Hancock-Beaulieu, M., Gull, A., and Lau, M. 1992. Okapi at TREC. In *TREC-1: Proceedings of the Text REtrieval Conference*. 21–30.

Singhal, A., Buckley, C., and Mitra, M. 1996. Pivoted document length normalization. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, New York, NY, USA, 21–29.

Smucker, M. D. and Allan, J. 2005. An investigation of dirichlet prior smoothing's performance advantage. Tech. Rep. IR-391, University of Massachusetts.

Turtle, H. and Flood, J. 1995. Query evaluation: strategies and optimizations. *Information Processing and Management 31,* 6, 831–850.

Tyree, S., Weinberger, K. Q., Agrawal, K., and Paykin, J. 2011. Parallel boosted regression trees for web search ranking. In *WWW '11: Proceedings of the 20th international conference on World wide web*. ACM, New York, NY, USA, 387–396.

van Rijsbergen, C. 1979. *Information Retrieval, 2nd edition*. Butterworths, London.

Vogt, C. 1997. When does it make sense to linearly combine relevance scores. In *SIGIR '97: Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, USA.

Vogt, C. C. and Cottrell, G. W. 1998. Predicting the performance of linearly combined ir systems. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, New York, NY, USA, 190–196.

Weinberger, K., Mohan, A., and Chen, Z. 2010. Tree ensembles and transfer learning. In *JMLR Workshop and Conference Proceedings*. Number 14. 1–24.

Wu, Q., Burges, C. J. C., Svore, K. M., and Gao, J. 2008. Ranking, boosting, and model adaptation. Tech. Rep. MSR-TR-2008-109, Microsoft.

Xu, J. and Li, H. 2007. Adarank: a boosting algorithm for information retrieval. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, New York, NY, USA, 391–398.

Zaragoza, H., Craswell, N., Taylor, M., Saria, S., and Robertson, S. 2004. Microsoft Cambridge at TREC-13: Web and HARD tracks. In *Proceedings of the 13th Text REtrieval Conference (TREC-2004)*. Gaithersburg, MD, USA.

Zhai, C. and Lafferty, J. 2001. A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, New York, NY, USA, 334–342.

Received 11th April 2012; revised 17th September 2012; revised 17 December 2012; accepted 6th February 2013.