Two-Stage Learning to Rank for Information Retrieval

Van Dang, Michael Bendersky, and W. Bruce Croft

Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts Amherst
{vdang, bemike, croft}@cs.umass.edu

Abstract. Current learning to rank approaches commonly focus on learning the best possible ranking function given a small fixed set of documents. This document set is often retrieved from the collection using a simple unsupervised bag-of-words method, e.g. BM25. This can potentially lead to learning a sub-optimal ranking, since many relevant documents may be excluded from the initially retrieved set. In this paper we propose a novel two-stage learning framework to address this problem. We first learn a ranking function over the entire retrieval collection using a limited set of textual features including weighted phrases, proximities and expansion terms. This function is then used to retrieve the best possible subset of documents over which the final model is trained using a larger set of query- and document-dependent features. Empirical evaluation using two web collections unequivocally demonstrates that our proposed two-stage framework, being able to learn its model from more relevant documents, outperforms current learning to rank approaches.

1 Introduction

Standard bag-of-words retrieval models such as BM25 or query likelihood have the advantage of being fast enough to be executed over an entire document index and yet effective enough to produce reasonably good results. However, these methods can only make use of a very limited number of features based on query term frequencies. In addition, the combination of these features is hard-coded into the retrieval model.

In contrast, learning to rank approaches [1] to information retrieval allow retrieval systems to incorporate hundreds or even thousands of arbitrarily defined features. Most importantly, these approaches automatically learn the most effective combination of these features in the ranking function based on the available training data. As a result, learning to rank approaches have consistently outperformed the standard bag-of-words retrieval models [2] [3].

However, due to the expense of computing a large number of arbitrary features, and the fact that many of these features are often not supported by the standard document indexing structures, learning to rank models are not applied to the entire document index. Instead, in current practice, learning to rank models operate in a two-stage fashion [1].

Fig. 1. An example query: “lower heart rate”. Higher recall at *Stage A* leads to better early precision at *Stage B*.

	Query	“lower heart rate”
	BM25	*MSE*
Relevant retrieved (*Stage A*)	30	73
NDCG@20 (*Stage B*)	19.88	58.88

At the first stage (*Stage A*), a simple bag-of-words model, e.g. BM25, is used to retrieve a small set of highly ranked documents from the entire document index. These retrieved documents, together with their human-assigned relevance labels, are then used to train a learning to rank model at the second stage (*Stage B*). At run-time, in response to user queries, the *Stage A* model is used again to retrieve a small set of highly ranked documents, which are then re-ranked by the *Stage B* model. Finally, the re-ranked results are presented to the user.

Given this re-ranking scheme, it is easy to see that while *Stage B* of the learning to rank framework should focus on high precision at the top ranks, especially for the purpose of web search, the model at *Stage A* should also aim for higher recall. This is due to the fact that if a relevant document is not retrieved at *Stage A*, it will never be surfaced to the top ranks at *Stage B*. Nevertheless, the majority of the current learning to rank literature focuses primarily on a variety of ways to improve the learned model at *Stage B*, while assuming that the model at *Stage A* is fixed to a standard bag-of-words model (most commonly, BM25) [4] [1]. This can potentially lead to learning a sub-optimal model, since many relevant documents might be excluded from the initially retrieved set.

As an intuitive example, consider the query “lower heart rate” in Figure 1. Figure 1 compares the performance of the learning to rank approach when for stage *Stage A* we use either (a) a simple BM25 method, or (b) a state-of-the-art query expansion method MSE [5]. Figure 1 shows that MSE retrieves twice as many relevant documents compared to BM25 at *Stage A*. This improvement in recall can be attributed, among other factors, to the fact that MSE uses expansion terms such as *hr, beta, block, exercise* and *bpm* to enhance the original query. The improvement in recall at *Stage A* results in a three-fold increase in an NDCG@20 metric at *Stage B*.

Following the example in Figure 1, the retrieval model at *Stage A* plays two important roles. The first role is to provide training data to train the model at *Stage B*, while the second role is to provide an initial pool of documents for this model to re-rank at run-time. Therefore, a more effective initial retrieval means both more relevant documents in the training data, as well as more relevant documents presented in the final ranking to the user.

Accordingly, in this paper, we propose an improved two-stage learning to rank framework. It modifies the existing two-stage approach by replacing the BM25 retrieval model at *Stage A* with a supervised learning approach that operates efficiently on a document index. In particular, we train the retrieval model at *Stage A* to improve the quality of the initial set of retrieved documents using a comprehensive set of textual features (which are available as a part of the standard indexing structures), including weighted phrases, proximities, and expansion terms.

While conceptually simple, the proposed change in the two-stage learning to rank approach is highly effective. Our experiments on two web collections, with both keyword and verbose queries, demonstrate that our framework significantly outperforms the existing approach, regardless of the learning to rank algorithms employed at *Stage B*. Our analyses confirm that using a better retrieval model at *Stage A* leads to a more effective model at *Stage B*, better initial document ranking, and consequently, better retrieval effectiveness. These results are consistent across collections and different query types.

2 Model

In this section, we describe the theoretical underpinning and the implementation details of the two-stage learning to rank framework. We begin by describing the general framework in Section 2.1. Then, in Section 2.2 and Section 2.3 we focus on the first and the second stages of the learning to rank process, respectively.

2.1 The Two-Stage Framework

Our proposed framework consists of two stages of learning. At *Stage A*, a ranker is trained on the entire document corpus to retrieve the most relevant set of documents from the collection. These documents are then used to train the ranker at *Stage B*. At run-time, the ranker at *Stage A* is used to retrieve a set of documents in response to a user query, which are then re-ranked by the ranker at *Stage B* to produce the final ranking for the user.

By design, the ranker at *Stage A* is recall-oriented and the ranker at *Stage B* is precision-driven. That is, the ranker at *Stage A* should be able to retrieve as many relevant documents as possible. If it fails to retrieve a relevant document, this document is neither annotated nor presented to the user at *Stage B*. Having more relevant documents retrieved at *Stage A* provides the ranker at *Stage B* with better data to learn from, as well as more potential for improving the final ranking presented to the user.

Formally, the two-stage learning to rank framework can be defined as follows. Let $T =< q_1, \ldots, q_m >$ be a training set of m queries and let $\mathcal{C} =< d_1, \ldots, d_n >$ be a retrieval corpus with n documents. Let M_A be a ranking function defined over the queries and the documents in the collections, such that

$$M_A : T \times \mathcal{C} \rightarrow \mathbb{R}, \tag{1}$$

A higher value of M_A indicates a higher likelihood of document relevance to the query.

Given some relevance metric of interest Λ (e.g., mean average precision), at *Stage A* we seek a ranker M_A^* such that

$$M_A^* = \arg\max_{M_A} \sum_{q \in T} \Lambda(M_A). \tag{2}$$

Once M_A^* is set (we will discuss the process of optimization of Equation 2 in the next section), we proceed to *Stage B*.

At *Stage B*, we seek to optimize ranker M_B, which is defined as

$$M_B\colon (q_i \in T) \times (d_j \in D^{(i)}_{M_A^*}) \rightarrow \mathbb{R}, \tag{3}$$

where $D^{(i)}_{M_A^*}$ is a set of k highest ranked documents retrieved by the initial ranker M_A^* in response to query q_i.

Similarly to the case of M_A, we seek to optimize the ranker M_B with respect to some relevance metric $\mathit{\Lambda}$ such that

$$M_B^* = \arg\max_{M_B} \sum_{q \in T} \mathit{\Lambda}(M_B, M_A^*). \tag{4}$$

Following Equation 4, the optimized ranker M_B^* has a dependency on the initial ranker M_A^*, since M_B is trained over a set of documents $D^{(i)}_{M_A^*}$ retrieved by the initial ranker.

Note that the two-stage learning to rank framework described in this section can be simply reduced to the standard learning to rank approach by setting M_A^* to be an unsupervised bag-of-words retrieval model such as BM25. Instead, in the next section we will explore several more effective alternatives for the ranking at *Stage A*.

2.2 Ranker M_A^*

Current applications of learning to rank for information retrieval [4], [1] commonly use standard unsupervised bag-of-words retrieval models such as BM25 as the initial ranking function M_A^*. However, recent research demonstrates that more complex retrieval models that incorporate phrases, term proximities and expansion terms can significantly outperform the standard bag-of-word models, especially in the context of large-scale web collections [6] [5] [7] [8] and longer, more complex queries [9] [10].

Accordingly, in this paper we adopt two state-of-the art supervised retrieval methods, `WSD` and `MSE`, as alternatives for the initial ranker M_A^*. Both of these methods incorporate textual features beyond query terms and were shown to be highly effective in prior work [6] [5].

Both of these methods are based on the *parameterized concept weighting* approach. They can incorporate arbitrary textual concepts (e.g., terms, phrases or term proximities) and assign weights to query concepts via a weighted combination of importance features (such as concept frequency in a large web collection or a Wikipedia title). Most generally, in the parameterized concept weighting approach, the ranker M_A is defined as follows:

$$M_A(q, d) \triangleq \sum_{\varphi \in \mathit{\Phi}} w_\varphi \sum_{c \in q} \varphi(c) sc(c, d), \tag{5}$$

where c are query concepts, Φ is a set of importance features associated with the query concepts, and $sc(c,d)$ is a frequency-based scoring function for concept occurrences in document d (e.g., BM25 or query-likelihood).

The weights w_φ of the importance features in Equation 5 are optimized using a learning to rank approach. This guarantees that the concept weights are assigned such that a particular relevance metric (e.g., mean average precision) is directly optimized. In this manner, WSD and MSE fit well in the two-stage learning to rank framework since, by optimizing Equation 5, they directly produce the most effective initial ranker M_A^* in Equation 2.

The methods WSD and MSE differ in their choice of the query concepts. While WSD uses only concepts that explicitly occur in the query, MSE also incorporates expansion terms from a variety of sources such as Wikipedia or an anchor text into the query formulation. Thus, the MSE retrieval model often leads to a higher relevant document recall and more diversity in the retrieved set [5]. For more details about these methods, the readers can refer to Bendersky et al. [6] [5].

Note that bigram statistics are the most expensive features used by WSD and MSE. They can be computed efficiently simply by building a bigram index. In addition, although MSE employs query expansion, the number of expansion terms is very small. As a result, both of these rankers are reasonably fast.

2.3 Ranker M_B^*

Given the rankings of documents retrieved by M_A^*, along with the human annotated relevance labels, the task of the ranker at *Stage B* is to learn a high-precision retrieval model M_B^* (see Equation 4). We adopt a standard learning-to-rank approach that is widely used in previous work [11] [12] [1] [4] [13].

Given a set of training queries $T =< q_1, \ldots, q_m >$ and a set $D^{(i)}_{M_A^*}$ of the k highest ranked documents by the initial ranker M_A^* for each query $q_i \in T$, the task is to learn a second ranker M_B^* to further optimize some relevance metric Λ (e.g., normalized discounted cumulative gain). In contrast to M_A^*, the ranker M_B^* is not evaluated over the entire corpus, but rather over a relatively small fixed set of documents. Therefore, it can make use of a larger set $F^{(i,j)} = \{f_1^{(i,j)}, f_2^{(i,j)}, ...\}$ of arbitrarily defined features

$$f^{(i,j)}\colon (q_i \in T) \times (d_j \in D^{(i)}_{M_A^*}) \rightarrow \mathbb{R},$$

over query-document pairs, without being prohibitively expensive to evaluate even for large-scale web collections.

The ranker in Equation 3 is then defined as a function of features in the set $F^{(i,j)}$

$$M_B(q_i, d_j) \triangleq g(F^{(i,j)}).$$

$g(F^{(i,j)})$ can be a linear combination of features (in linear models) or some form of regression trees (in tree-based models).

To ensure a state-of-the-art effectiveness of ranker M_B^*, in this paper we implement a set of features $F^{(i,j)}$, which incorporates a wide range of features

Table 1. Set of features $F^{(i,j)}$, used in the second stage by the M_B^* ranker

Feature	**Document Section**
TF, IDF, TF*IDF (min/max/sum/mean/var)	[Body, Anchor, Title, Whole page]
Number of covered query terms	All
Document length	All
BM25	All
Query Likelihood (Two-stage/Dirichlet/JM smoothing)	[Body, Anchor, Title, All]
Sequential Dependence (Two-stage/Dirichlet/JM smoothing)	[Body, Anchor, Title, All]
URL Length/Depth	
Number of in-links	
PageRank	
Stopwords fraction/coverage	All
Number of terms/Term entropy	All
Score from M_A^*	All

used in the previous work on learning to rank [4] [14] [15] [7]. Table 1 provides an overview of the implemented features.

Several learning to rank algorithms have been proposed to find the optimal ranker M_B^* in Equation 4. They can be categorized into three approaches: point-wise, pair-wise and list-wise. While the former two have the relevance measure Λ built-in, the latter allows this function to be arbitrarily defined. Liu [1] provides a good overview of various learning to rank methods. In this paper, we evaluate the most competitive algorithms from each of the three classes above:

- **MART** [16] is a state-of-the-art regression model, which is an ensemble of regression trees constructed using the boosting approach. Using regression models for learning to rank has been known as the point-wise approach.
- **RankBoost** [17] is a pair-wise learning approach based on *AdaBoost*. It learns a linear combination of weak rankers that minimizes pair-wise loss. Each weak ranker consists of a single feature and a threshold that best distinguishes between relevant and non-relevant documents. RankBoost has been popular baseline in the learning to rank community.
- **Coordinate Ascent** [2] is a list-wise algorithm that can optimize any IR measures directly. It cycles through each of the features and optimizes over it while holding the others fixed until no more improvement is observed.
- **LambdaMART** [18] is the winning approach at the Yahoo! Learning to Rank Challenge [19]. LambdaMART is derived from LambdaRank [12], which uses neural networks to minimize pair-wise cost, scaled with the amount of change in the target measure incurred when swapping these two documents. This scaling approach has proven to be equivalent to list-wise optimization [20]. Since LambdaMART uses MART to minimize LambdaRank scaled pair-wise cost, it is essentially a list-wise method that works in a pair-wise fashion.

3 Related Work

The current learning to rank approach can be formulated as a two-stage process. An initial model (*Stage A*) is first used to retrieve a sample of documents from the entire collection index. A second model (*Stage B*) is used to re-rank these documents before presenting them to users.

In this two-stage re-ranking scheme, it is critical for the *Stage A* model to have a good coverage of relevant documents. The literature, however, has been concentrating primarily on developing more powerful models for the second stage, while using simple bag-of-words models in the first stage [4] [1]. In contrast, our focus in this paper is on improving the recall of the *Stage A* model with two state-of-the-art learning techniques: `WSD` [21] and `MSE` [5]. The example in Figure 1 demonstrates the advantage of having a higher initial coverage, which is akin to the situation in the pseudo relevance feedback, where an initial set of retrieved documents for a query is used to modify the query and produce a new ranking. As in the case of pseudo relevance feedback [22], we show that in the learning to rank setting a better initial ranking produces a better final ranking.

There have been some studies on the effectiveness of the *Stage A* model. Aslam et al. [23] have studied several document sampling strategies and shown that some of them can improve the quality of the initial retrieved set over BM25. These methods, however, are intended to be used during training only since some of them do not aim to retrieve more relevant documents but rather those that are "interesting" to the learning process. Donmez and Carbonell [24] propose to further sub-sample the initial sample of documents using active learning in order to focus the learning process on the most informative training instances, which is again on the learning side of the framework. Our *Stage A* model, on the contrary, aims to provide more relevant documents not only for the *Stage B* model to learn from (training) but also to re-rank (run-time).

In addition, researchers have also examined how learning to rank models (*Stage B*) are affected by different characteristics of the training data, such as the sample size and the similarity among documents used for training [23] [13], as well as different training metrics [20] [25] [13]. Since these studies are independent of the methods used for retrieving the documents in the training set, their findings should also apply in our framework.

As we described earlier, any learning to rank algorithm [17] [2] [11] [12] can be applied in the second stage of our framework. These algorithms can be broadly classified into three approaches: point-wise, pair-wise and list-wise. The point-wise approach attempts to accurately predict the relevance label for individual documents. Pair-wise methods focus instead on the ability to rank relevant documents higher than the non relevant. List-wise techniques take the entire ranked list as input and directly optimize retrieval measure defined upon this list. Further details can be found in [1]. In our experiments, we consider four popular algorithms across three classes: *MART* [16] (point-wise), *RankBoost* [17] (pair-wise), Coordinate Ascent [2] and *LambdaMART* [18] (list-wise).

4 Experimental Setup

Our retrieval experiments are conducted on two TREC web collections: *Gov2* and *ClueWeb-B*. *Gov2* is a collection of web pages from the *.gov* domain crawled in 2004. *ClueWeb-B* is the first segment of a larger web corpus (ClueWeb-A) created in 2009. The corpus statistics are provided in Table 2.

Table 2. Summary of the *Gov2* and *ClueWeb-B* collections.

Name	#Docs	Topic Numbers
Gov2	25,205,179	701-850
ClueWeb-B	50,220,423	1-100

We use two types of queries for evaluation. The first type are short keyword queries (TREC topic titles), while the second type are verbose natural language queries (TREC topic descriptions). Indri/Lemur [1] is used to build indexes and perform retrieval experiments. At indexing time, all documents are stemmed using Krovetz stemmer. The Dirichlet smoothing parameter μ is set to 2500 (the default Indri configurations). Stop-words removal are done only at query time using the standard INQUERY stop list. All statistical significance tests are performed using Fisher's randomization test with 20,000 iterations and $\alpha < 0.05$.

As mentioned in Section 2, we adopt the state-of-the-art *Weighted Sequential Dependence* (`WSD`) and *Multiple Source Expansion* models (`MSE`) as our *Stage A* model. `WSD` and `MSE` go beyond bag-of-words by representing each query using not only explicit concepts from the query string but also latent concepts obtained via pseudo relevance feedback. The weights for each of these concepts are learned automatically using parametrized combination of importance features. As candidates for *Stage B* models, we employ four competitive learning to rank algorithms, namely *MART* [16] (list-wise), *RankBoost* [17] (pair-wise), *Coordinate Ascent* [2] and *LambdaMART* [18] (list-wise). Following some studies on the robustness of measures that are based on the entire ranked list [25], mean average precision (MAP) is the optimization in both stages.

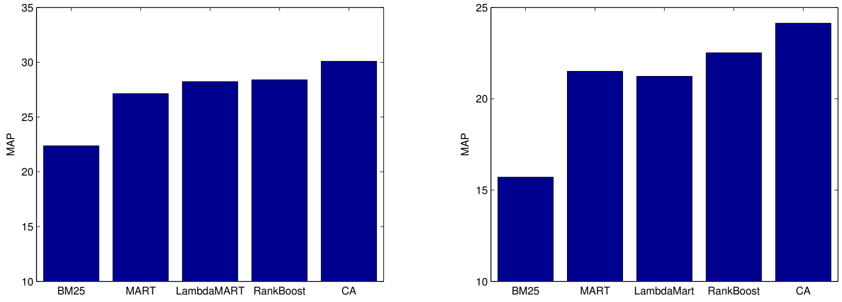
5 *Stage B* Evaluation

Though learning to rank algorithms proved to be effective in general [1], their success on TREC data has been limited. In the Web Track 2011, for example, while McCreadie et al. [14] achieved good results, Boytsov and Belova [26] found that their learning to rank models do not provide consistent and substantial improvements over a hand-tuned formula.

In order to study the impact of the *Stage A* model, we need the *Stage B* model to be reasonably effective. Therefore, in this section we first study the performance of the four candidate learning to rank algorithms. For this purpose, we fix the *Stage A* model to be `BM25`. All the experiments are conducted using 3-fold cross-validation. In each fold, the training data is further split into two: 70% for training and 30% for validation. Fig. 2 (a) and (b) provides the results on *Gov2* and *ClueWeb-B* respectively.

[1] http://www.lemurproject.org/

Fig. 2. Performance of the four learning to rank models.



(a) *Gov2* ⟨*title*⟩

(b) *ClueWeb-B* ⟨*title*⟩

Our results show that all four algorithms substantially outperform the `BM25` baseline. This affirms the effectiveness of these algorithms, as well as the feature set in Table 1. It is worth noting that our experiments only intend to make sure our candidate *Stage B* models are indeed effective, but not to fully compare the four algorithms, which would require evaluation on multiple datasets with different characteristics.

Not surprisingly, both MART and LambdaMART are the least effective algorithms, since tree-based models often require large number of training queries to avoid over-fitting. As evidence, they are the top performing approaches on the Yahoo! Learning to Rank data [19], which contains about $10K$ queries and 700 features. In contrast, our collections have less than 200 queries with about 100 features.

6 *Stage A* Evaluation

6.1 Effectiveness of the Two-Stage Learning to Rank

In this section, we evaluate the three *Stage A* models, namely `BM25`, `WSD` [21] and `MSE` [5] with the *Stage B* model fixed to one of the four learning to rank algorithms. Due to space limitations, we only present the results with Coordinate Ascent (`CA`), simply because it is the best performing approach on our data. Results with the other three algorithms, in fact, lead to the same conclusions.

Learning in both stages is also done with 3-fold cross-validation. Let us use `WSD`/`CA` as the *Stage A*/*Stage B* models to explain the learning process. In each fold, we first train `WSD` from the queries specified by the training data, using documents from the entire retrieval collection. After that, this model is used to retrieve the top 1000 documents for these queries, which are used to train a `CA` model, marked as `CA[WSD]`. To evaluate this model, `WSD` is again used to retrieve the top 1000 documents for the test queries, which are re-ranked by `CA[WSD]` to produce the final ranking. Test results on both collections, and for both types of

Table 3. Performance comparison among three *Stage A* models – BM25, WSD and MSE – with the *Stage B* model fixed to CA. Statistical significant differences are marked by $*$.

Gov2				
	⟨*title*⟩		⟨*description*⟩	
	NDCG@20	MAP	NDCG@20	MAP
CA[BM25]	47.80	30.09	40.69	26.40
CA[WSD]	48.24 *(+0.92%)*	34.26* *(+13.86%)*	43.87* *(+7.82%)*	29.93* *(+13.37%)*
CA[MSE]	50.19* *(+5.0%)*	36.12* *(+20.0%)*	45.27* *(+11.3%)*	32.41* *(+22.7%)*

ClueWeb-B				
	⟨*title*⟩		⟨*description*⟩	
	NDCG@20	MAP	NDCG@20	MAP
CA[BM25]	28.36	24.14	22.07	15.32
CA[WSD]	30.93* *(+9.06%)*	25.58 *(+5.97%)*	22.17 *(+0.45%)*	15.90 *(+3.79%)*
CA[MSE]	32.20* *(+13.5%)*	27.19* *(+12.6%)*	24.65* *(+11.7%)*	16.74* *(+9.2%)*

Table 4. Effectiveness of each of the three *Stage A* models in Recall and MAP. Statistical significant differences are marked by $*$.

Gov2				
	⟨*title*⟩		⟨*description*⟩	
	Recall	MAP	Recall	MAP
BM25	59.49	22.35	59.82	23.28
WSD	74.56* *(+25.3%)*	31.60* *(+41.4%)*	68.03* *(+13.7%)*	28.21* *(+21.2%)*
MSE	75.85* *(+27.5%)*	34.28* *(+53.4%)*	69.66* *(+16.4%)*	30.83* *(+32.4%)*

ClueWeb-B				
	⟨*title*⟩		⟨*description*⟩	
	Recall	MAP	Recall	MAP
BM25	66.81	15.72	50.50	11.46
WSD	73.13* *(+9.5%)*	18.36* *(+16.8%)*	55.02* *(+9.0%)*	13.75* *(+20%)*
MSE	75.37* *(+12.8%)*	22.33* *(+42.1%)*	58.11* *(+15.1%)*	15.35* *(+34%)*

queries, are reported using MAP and NDCG@20. Experiments with BM25 and MSE are done similarly except that BM25 requires no training in the first stage.

Table 3 presents the results obtained with CA as the *Stage B* model. It is obvious that CA with WSD and MSE as the *Stage A* models substantially outperforms the existing approaches (BM25). This is consistent across query types and collections, confirming the importance of the *Stage A* model. MSE is the most effective *Stage A* model. CA[MSE] achieves an average gain of 16% in MAP over the CA[BM25] baseline.

6.2 Analysis

To understand why CA[MSE] and CA[WSD] are more effective than CA[BM25], we examine Recall and MAP of the three *Stage A* models in Table 4. It is clear that both WSD and MSE consistently outperform BM25 in both measures, indicating that the former two methods provide more relevant documents for CA to learn from (training) and to re-rank (test). In particular, MSE is the best performing methods in both Recall and MAP, which confirms the importance of query expansion at *Stage A*.

Given the results in Table 4, is the superior end-to-end performance of CA[WSD] and CA[MSE] due to the fact that they enable better learning at *Stage*

B? Or do they perform better simply because they have more relevant documents to start with? To answer this question, we apply the models `CA[WSD]` and `CA[MSE]` to *re-rank documents retrieved by* `BM25`. We denote these re-ranking models as `CA[WSD][BM25]` and `CA[MSE][BM25]`, respectively.

Table 5 reveals that both `CA[WSD][BM25]` and `CA[MSE][BM25]` achieve significantly higher $NDCG@20$ compared to `CA[BM25]`. However, the gains in MAP are not as significant. This demonstrates that both a higher precision model and a better relevant document coverage contribute to the substantial end-to-end improvements presented in Table 3.

Table 5. Performance of our models on documents retrieved by `BM25`. Statistical significant differences are marked by $*$.

Gov2				
	⟨*title*⟩		⟨*description*⟩	
	NDCG@20	MAP	NDCG@20	MAP
CA[BM25][BM25]	47.80	30.09	40.69	26.40
CA[WSD][BM25]	48.20 *(+0.8%)*	30.27 *(+0.6%)*	42.69* *(+4.9%)*	27.56* *(+4.4%)*
CA[MSE][BM25]	50.29* *(+5.2%)*	31.54* *(+4.8%)*	44.57* *(+9.5%)*	29.81* *(+12.9%)*

ClueWeb-B				
	⟨*title*⟩		⟨*description*⟩	
	NDCG@20	MAP	NDCG@20	MAP
CA[BM25][BM25]	28.36	24.14	22.07	15.32
CA[WSD][BM25]	30.07* *(+6.0%)*	24.48 *(+1.4%)*	22.44 *(+1.7%)*	15.59 *(+1.8%)*
CA[MSE][BM25]	30.74* *(+8.4%)*	25.34* *(+4.9%)*	23.38*(+5.9%)*	15.47*(+1.0%)*

7 Conclusions

Learning to rank has been studied as a two-stage process where an initial ranker (*Stage A*) retrieves a set of documents and a second model (*Stage B*) re-ranks them before presenting to the users. The role of the initial ranker is very important, yet often overlooked. Existing work usually deploys a simple bag-of-words model such as BM25 at *Stage A* and focuses instead on developing complex models for *Stage B*. In this paper, we show that using better models at *Stage A* is a simple way of significantly improving the retrieval effectiveness of learning to rank. We empirically demonstrate that our approach helps (1) to train a more effective learning to rank model for *Stage B* and (2) to provide more room for the *Stage B* model to improve the final ranking. The resulting effectiveness improvements are consistent across different collections and query types.

References

1. Liu, T.Y.: Learning to rank for information retrieval. Foundations and Trends in Information Retrieval **3**(3) (2009) 225–331
2. Metzler, D., Croft, W.B.: Linear feature-based models for information retrieval. Information Retrieval **10**(3) (2007) 257–274

3. Xu, J., Li, H.: Adarank: a boosting algorithm for information retrieval. In: SIGIR. (2007) 391–398
4. Liu, T.Y., Xu, J., Qin, T., Xiong, W., Li, H.: LETOR: Benchmark Dataset for Research on Learning to Rank for Information Retrieval. In: SIGIR. (2007)
5. Bendersky, M., Metzler, D., Croft, W.B.: Effective query formulation with multiple information sources. In: WSDM. (2012) 443–452
6. Bendersky, M., Metzler, D., Croft, W.B.: Learning concept importance using a weighted dependence model. In: WSDM. (2010) 31–40
7. Metzler, D., Croft, W.B.: A Markov random field model for term dependencies. In: SIGIR. (2005) 472–479
8. Peng, J., Macdonald, C., He, B., Plachouras, V., Ounis, I.: Incorporating term dependency in the DFR framework. In: SIGIR. (2007) 843–844
9. Lease, M.: An improved markov random field model for supporting verbose queries. In: SIGIR. (2009) 476–483
10. Lu, Y., Peng, F., Mishne, G., Wei, X., Dumoulin, B.: Improving Web search relevance with semantic features. In: EMNLP. (2009) 648–657
11. Burges, C.J.C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G.N.: Learning to rank using gradient descent. In: ICML. (2005) 89–96
12. Burges, C.J.C., Ragno, R., Le, Q.V.: Learning to Rank with Nonsmooth Cost Functions. In: NIPS. (2006) 193–200
13. Macdonald, C., Santos, R., Ounis, I.: The whens and hows of learning to rank for web search. Information Retrieval (2012) 1–45
14. McCreadie, R., Macdonald, C., Santos, R.L.T., Ounis, I.: University of glasgow at trec 2011: Experiments with terrier in crowdsourcing, microblog, and web tracks. In: TREC. (2011)
15. Bendersky, M., Croft, W.B., Diao, Y.: Quality-biased ranking of web documents. In: WSDM. (2011) 95–104
16. Friedman, J.H.: Greedy function approximation: A gradient boosting machine. Annals of Statistics **29** (1999) 1189–1232
17. Freund, Y., Iyer, R., Schapire, R., Singer, Y.: An efficient boosting algorithm for combining preferences. The Journal of Machine Learning Research **4** (2003) 933–969
18. Q. Wu, C.J.C. Burges, K.S., Gao, J.: Adapting boosting for information retrieval measures. Information Retrieval **13**(3) (2010) 254–270
19. O. Chapelle, Y.C.: Yahoo! learning to rank challenge overview. Machine Learning Research - Proceedings Track **14** (2011) 1–24
20. Donmez, P., Svore, K.M., Burges, C.J.C.: On the local optimality of LambdaRank. In: SIGIR. (2009) 460–467
21. Bendersky, M., Metzler, D., Croft, W.B.: Parameterized concept weighting in verbose queries. In: SIGIR. (2011) 605–614
22. Metzler, D., Croft, W.B.: Latent concept expansion using markov random fields. In: Proceedings of the Annual ACM SIGIR Conference. (2007) 311–318
23. Aslam, J.A., Kanoulas, E., Pavlu, V., Savev, S., Yilmaz, E.: Document selection methodologies for efficient and effective learning-to-rank. In: SIGIR. (2009) 468–475
24. Donmez, P., Carbonell, J.G.: Active sampling for rank learning via optimizing the area under the roc curve. In: ECIR. (2009) 78–89
25. Yilmaz, E., Robertson, S.: On the choice of effectiveness measures for learning to rank. Information Retrieval **13** (2010) 271–290
26. Boytsov, L., Belova, A.: Evaluating learning-to-rank methods in the web track adhoc task. In: TREC. (2011)