Senior Thesis

An Experimental Study of Multi-stage Retrieval Systems

Mohammed Yusuf Ansari

Advisor: Mohammad Hammoud

April 2020

Abstract

Information Retrieval (IR) is concerned with searching over large unstructured data like web pages, emails, and image libraries, among others. This large-scale searching is made possible through IR systems, which pre-process and store such data in a space- and time-efficient structure known as an *inverted index*. In an IR system, the process of retrieving relevant data (referred, henceforth, to as documents) for a given user query passes through three major stages, namely: (1) *candidate set generation*, (2) *feature extraction*, and (3) *candidate set re-ranking* stages [1].

During the candidate set generation stage (or stage 1), a retrieval strategy with a pruning model (e.g., WAND [2] and MaxScore [3]) is executed on top of an inverted index to retrieve top-K ranked documents for any given user query. In the feature extraction stage (or stage 2), various features are extracted from the top-K documents generated by stage 1. In the candidate set re-ranking stage (or stage 3), the features extracted by stage 2 are dispatched to a trained machine learning model, which re-ranks the top-K documents accordingly. After re-ranking, the top 10-50 documents are output to the user as a final ranked list.

In this work, we discovered that the effectiveness of machine learning models in stage 3 relies heavily on the configuration parameters defined in stages 1 and 2. Nonetheless, these machine learning models are typically developed and trained in complete obliviousness to such parameters, yielding thereby a significant loss in end-to-end effectiveness of IR systems.

To this end, we thoroughly investigated the correlation between the different stages of real-world end-to-end multi-stage IR systems to understand how they impact each other and, accordingly, recoup lost effectiveness. In particular, we asked and answered two critical research questions. First, to what extent do stages 1 and 2 influence the effectiveness of stage 3? Second, what parameter values and machine learning models should we use in stages 1, 2, and 3 to achieve high effectiveness in end-to-end IR systems?

We answered the above two critical questions through a comprehensive experimental study that observed, identified, and explained the effectiveness of a representative search engine under various configuration parameters and machine learning models. Subsequently, we provided recommendations for improving the effectiveness of real-world IR systems and suggested promising research directions.

Chapter 1

Introduction

It is critical for an Information Retrieval (IR) system to be *efficient* (i.e., quick) and *effective* (i.e., accurate). For instance, search engines like Google, Bing and Baidu are deemed as popular applications of *efficient* and *effective* multi-stage IR systems. In general, if a search engine consumes noticeable time to respond to users or returns irrelevant documents, it will be deemed inefficient or ineffective in meeting user expectations.

In order to achieve desired levels of effectiveness and efficiency, modern IR systems encompass three stages, namely: (1) *candidate set generation*, (2) *feature extraction*, and (3) *candidate set re-ranking* stages [1]. In the candidate set generation stage (or stage 1), a retrieval strategy (e.g., MaxScore [3] and LazyBM [4]) and a ranking function are executed on top of an inverted index to retrieve top-K ranked documents. In the feature extraction stage (or stage 2), various features are extracted from the top-K documents returned by stage 1. In the candidate set re-ranking stage (or stage 3), the features extracted by stage 2 are conveyed to a trained machine learning model, which re-ranks the top-K documents accordingly.

In an attempt to improve the efficiency of IR systems, several retrieval strategies that involve sophisticated pruning models were developed over the past few years [2, 3, 4]. Researchers have also created machine learning models to predict the optimal size of a candidate set using pre-retrieval static features [5]. These retrieval strategies and machine learning models served significantly in increasing the efficiency of stage 1, while maintaining overall effectiveness. To boost effectiveness, other machine learning models like Gradient Boosted Regression Trees (GBRT) [6] and LambdaMART [7] were proposed for stage 3 and extra features (e.g., DPH-DFR [8]) were added to stage 2 [9].

In this work, we observed that these machine learning models were developed in complete obliviousness to the configuration parameters (e.g., size of the candidate set, types of features, etc.,) of stages 1 and 2. In particular, we realized that the number of relevant documents retrieved (i.e., *recall*) and the size of the candidate set (K) in stage 1 highly impact the performance of the machine learning models in stage 3. Moreover, we noticed that the type and number

of features extracted in stage 2 considerably influence the effectiveness of stage 3 because different machine learning models perform differently with different input features. As such, we suggest that optimizing configuration parameters in stage 1 or stage 2 without considering the impact on stage 3 may yield sub-optimal end-to-end effectiveness for multi-stage IR systems.

Consequently, we propose that examining and capitalizing on the performance correlation between the different stages of IR systems can serve in tremendously improving effectiveness. To our knowledge, this performance correlation and its real impact on user experience have not been fully investigated in literature. This thesis aims at filling this critical gap through an extensive empirical investigation, which observes, identifies, and sets optimal configuration parameters in stages 1 and 2 for empowering state-of-the-art machine learning models in stage 3 and increasing end-to-end effectiveness.

To elaborate, we conducted comprehensive experimental studies for identifying optimal configuration parameters in different stages of multi-stage IR systems. Specifically, we studied stage 1 in-depth (i.e., *vertically*) to analyze the recall and nature of top-K documents. Afterwards, we tapped into the results of this study to explore stages 2 and 3 *horizontally* (i.e., by considering how they impact each other) and evaluate the performance of different features. Finally, we utilized the insights garnered from these vertical and horizontal analyses to explain and foster the effectiveness of machine learning models and, subsequently, end-to-end IR systems.

In our experimental studies, we used the standard ClueWeb12B document collection and TREC Webtrack 2013 query set. In addition, we leveraged and stitched together different standard frameworks that represent the different stages of end-to-end IR systems.

To summarize, this thesis proposes and answers the following research questions using an experimental approach:

- What is the performance correlation between the number of relevant documents (i.e., *recall*) and the size of the candidate set (i.e., K)?
- What is the best set of features (i.e., *query-dependent*, *static-document*, or *pre-retrieval*) for machine learning models in stage 3? Is there a combination of features that works best for machine learning models?
- What is the best size of the candidate set (K) for machine learning models in an end-to-end IR system?
- Which machine learning model maximizes the end-to-end effectiveness of IR systems the most?
- How do machine learning models perform on different subsets of query-dependent features (e.g., BM25 [10], LMDIR [11], etc.,)?

The rest of the thesis is organized as follows. A summary of related work is provided in Chapter 2. Details of experimental methodology are presented in Chapter 3. We demonstrate and discuss our experimental studies in Chapter

4. Based on our experimental findings, we recommend optimal configuration parameters and machine learning models in Chapter 5. Finally, we conclude with few remarks and suggest promising research directions in Chapter 6.

Chapter 2

Related Work

There has been a significant effort in the past few years to increase the efficiency of the candidate set generation stage and the effectiveness of the candidate set re-ranking stage. Researchers have optimized parameters in each stage and measured the resultant impact on the effectiveness and efficiency of the underlying end-to-end retrieval system. We next summarize some of the work that has been conducted in the field.

2.1 Impact of retrieval strategies on effectiveness:

Lin et al. [12] analyzed effectiveness/efficiency tradeoffs with three candidate set generation approaches: postings intersection with SvS [12], conjunctive query evaluation with WAND [2], and disjunctive query evaluation with WAND as well. Fixed configurations of the feature extraction and candidate set re-ranking stages were used to isolate the end-to-end effectiveness and efficiency implications of different algorithms. Based on their experiments, N. Asadi and J. Lin concluded that postings intersection with SvS is the most efficient candidate set generation approach. However, SvS was shown to have low end-to-end effectiveness as compared to conjunctive and disjunctive WAND. This result highlights the importance of term and document frequencies for the candidate set generation stage. Independently, the paper also concluded that conjunctive WAND is a better candidate set generation algorithm because it is much faster than disjunctive WAND in query evaluation. Moreover, in terms of end-to-end effectiveness, it is statistically distinguishable from disjunctive WAND.

2.2 Predicting the size of the candidate set (K):

Culpepper et al. [5] based their work on the fact that the final ranked list of documents is relatively insensitive to the quality of the initial candidate set in terms of early precision. They realized that if the candidate set has some good quality documents then the machine learning model in the candidate set re-

ranking stage can identify and re-rank them high. Using this idea, they trained a cascade of binary random forest classifiers based on seventy pre-retrieval static features (e.g., term frequency, document frequency, etc.,). The classifier predicts the best value of the candidate set size (K) on a query-by-query basis, while minimizing the effectiveness loss and maximizing the efficiency gain. This idea of predicting optimal K on per query basis is beneficial because it allows the candidate set generation stage to potentially retrieve fewer documents. As a byproduct, this will decrease the cost of extracting document features in the feature extraction stage, thus, serving in maximizing efficiency. The binary cascaded classification is able to achieve up to a 50 % reduction in average candidate set size.

2.3 Quality vs efficiency in learning-to-rank models:

Capannini et al. [13] investigated the training parameters of different machine learning algorithms and their impact on the quality versus efficiency tradeoff. Experiments concluded that there is no overall best algorithm, but the optimal choice depends on the time budget available for the multi-stage retrieval systems. More precisely, the paper analyzes four broad families of learning-to-rank algorithms, namely: (1) *linear combinations*, (2) *artificial neural networks*, (3) *forests of regression trees*, and (4) *forests of oblivious trees*. The performance of each algorithm was plotted on a quality-cost curve, where quality was measured using NDCG (standard effectiveness metric in IR) and cost was quantified as the time taken to score the documents. Using these quality-cost curves, authors were able to recommend the best algorithm for a given time budget. The paper also suggested that for a maximum time budget of 100 microseconds, the LTR algorithms LambdaMART [7] and MART [6] are most effective.

2.4 Early exit for additive machine learning models:

Combazoglu et al. [14] proposed four different optimization strategies that allow short-circuiting score computation in additive machine learning systems. The optimization strategies are: (1) *early exit using score threshold* (EST), (2) *early exit using capacity threshold* (ECT), (3) *early exit using rank threshold* (ERT), and (4) *early exit using proximity threshold* (EPT). In EST, exits are based on comparisons between accumulated scores and offline-computed score thresholds. In ECT, exit decisions are based on a partial set of previously seen document scores. This is done by maintaining a maximum score heap. Afterwards, if the accumulated score of a document is less than the minimum score in the heap, the document score computation is short circuited. Early exits decisions in ERT are based on comparisons between the current ranks of documents (computed over all documents) and offline-computed rank thresholds. EPT utilizes the idea of scoring the documents that have scores close to the score of the document at the K^{th} rank. These strategies are able to expedite the score computations by more than four times with almost no loss in result quality [4].

2.5 Learning to rank data sets (LETOR 4.0):

Tao Qin et al. [15] made data sets for research on learning-to-rank, which contains standard features, relevance judgements, evaluation tools and several baselines. The data sets were constructed using the Gov2 web page collection (∼25 million documents) and two query sets from Million Query track of TREC 2007 (MQ2007) and TREC 2008 (MQ2008). All the data sets are designed to be run with 5-fold cross validation strategy. In each fold, three subsets are used for training, one subset is used for validation and the remaining one is used for testing. LETOR 4.0 data sets can be used for addressing the following problems in learning-to-rank: (1) *supervised learning*, (2) *semi-supervised learning* and (3) *rank aggregation*. In supervised learning, machine learning algorithms re-rank based on judged query-document feature vectors. On the other hand, semi-supervised learning differs from supervised learning because it involves both judged and unjudged query-document pairs. In rank aggregation, a query is associated with a set of ranked list. The task of is to output a better ranked list by aggregating multiple input lists. Along with the data sets, LETOR 4.0 also provides meta data for all queries, link graph of Gov2 collection, sitemap of Gov2 collection and a detailed list of features used in creating the data sets.

2.6 Cascade ranking model for efficient ranked retrieval:

Lin et al. [16] tried to analyze tradeoff between effectiveness and efficiency when designing retrieval models for large scale document collection. Generally, effectiveness is derived from complex ranking functions, such as those utilized in the learning-to-rank models. On the other hand, efficiency is enhanced through a variety of approaches such as index pruning, feature pruning, approximate query evaluation, and systems engineering. Factors affecting effectiveness and efficiency are inherently disjoint in nature making it difficult to jointly optimize them in a multi-stage IR system. To address this problem, Lin et al. [16] developed a novel cascade ranking model for efficient ranked retrieval. The cascade uses a sequence of increasingly complex ranking functions to progressively prune documents and refine the rank order of the non-pruned documents. Thus, the cascade model views retrieval as a multi-stage refinement problem. This paradigm is well suited for large document collections, because the number of documents relevant to a given query is very small as compared to collection size. Authors also introduce a novel boosting algorithm that jointly learns the model structure and the pruning criteria at each stage within the cascade. Experiments conducted on Wt10g, Gov2 and Clue document collections show that the cascade model is able to simultaneously achieve high effectiveness and fast retrieval.

Chapter 3

Experimental Methodology

In this section, we describe our experimental methodology in terms of the frameworks used to simulate the different stages of an end-to-end IR system as well as the corpus and queries utilized for conducting experiments. Furthermore, we explain how we trained our machine learning models and measured their effectiveness.

3.1 Frameworks

There are two major objectives of any search engine: (1) *effectiveness* (i.e., accuracy), (2) *efficiency* (i.e., speed). As discussed earlier, in order to meet effectiveness and efficiency goals, multi-stage IR systems typically involve three stages, namely: (1) *candidate set generation stage*, (2) *feature extraction stage*, and (3) *candidate set re-ranking stage*. We next elaborate on the three frameworks we used for simulating the three stages of IR systems.

As shown in Figure 3.1, in the candidate set generation stage, a retrieval strategy (e.g., WAND [2]) with a ranking function (e.g., BM25 [10]) is executed on top of an inverted index to retrieve top-K ranked documents for given user queries. We are using the PISA engine framework [17] to simulate the functionality of the candidate set generation stage. PISA is a text search engine written in C++ for enhanced performance. It allows working on a document collection from scratch by providing indexing, parsing, and sharding capabilities [17]. For instance, it involves a parse_collection utility, which enables users to parse the document collection into a *forward index*. A forward index is a data structure that stores the term IDs associated with every document in the collection. Later, the forward index can be converted into an *inverted index*, which can be used for query evaluation. An inverted index stores for each unique term the document IDs of the documents in which the term appears.

In addition, PISA provides implementations for many compression techniques, which allow compressing the posting lists of an inverted index. Some of the compression techniques are *Binary Interpolative Coding* [18],

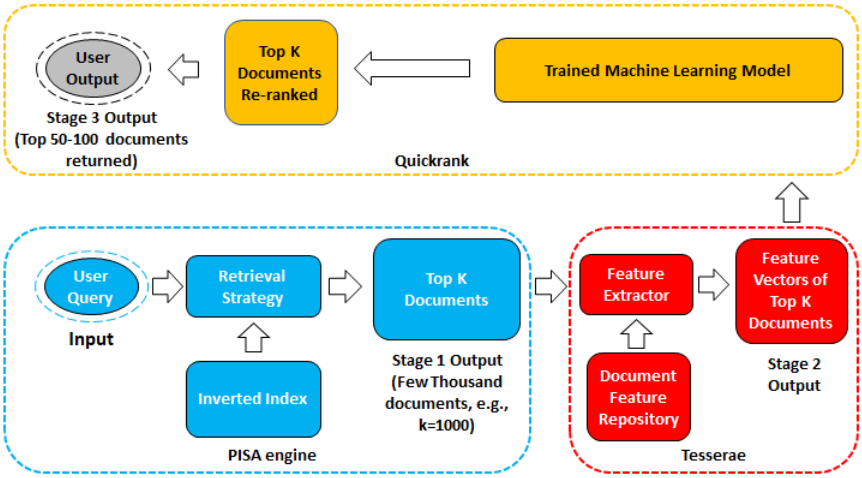


Figure 3.1: Frameworks

Elias-Fano [19], *MaskedVByte* [20], *OptPFD* [21], *QMX* [22], and *SIMD-BP128* [23]. A compressed inverted index can then be used for query evaluation. PISA incorporates various query evaluation algorithms, including *AND*, *OR*, *MaxScore* [3], *WAND* [2], *BlockMax WAND* [24], *BlockMax MaxScore*, and *Variable BlockMax WAND* [25]. In summary, PISA provides all the necessary functionality required for simulating the candidate set generation stage of a multi-stage IR system.

The next stage in the pipeline of a multi-stage IR system is the feature extraction stage as illustrated in Figure 3.1. In this stage, various features of the top-K documents (which were returned by the first stage) are extracted. We used the Tesserae framework [26] as a feature extractor for query-document pairs. In order to extract query-document features, Tesserae needs to build its own index using Indri [27]. After the Tesserae index is built, it can be used to extract three different types of features: (1) *query-dependent features*, (2) *static-document features*, and (3) *unigram pre-retrieval features*. Once the features are extracted, they can be fed into a machine learning framework for re-ranking.

The final stage in a multi-stage IR system is candidate set re-ranking as demonstrated in Figure 3.1. We use the Quickrank framework [13] for re-ranking document lists with different machine learning algorithms. Quickrank is written in C++ so as to achieve high efficiency. Quickrank provides a diverse selection of learning-to-rank algorithms for re-ranking document lists (e.g., *GBRT* [6], *LamdaMART* [7], *Oblivious GBRT/LamdaMART* [28], *CoordinateAscent* [29], *LineSearch* [30], *RankBoost* [31], and *DART* [32]). In addition, it encompasses novel learning optimizations like *CLEAVER* [33], *X-CLEAVER* [34], and *X-DART* [35]. In short, Quickrank provides all the necessary tools required for generating a final ranked list of documents.

3.2 Benchmarks

We conducted our experiments using the large ClueWeb12 category B corpus, which consists of 52 million documents. ClueWeb12 is a successor of ClueWeb09 corpus, which was crawled in early 2012. The crawl was initially seeded with 2,820,500 unique URLs [36]. This list was generated by taking the 10 million ClueWeb09 URLs that had the highest PageRank scores, and then removing any page that was not in the top 90% of pages as ranked by Waterloo spam scores (i.e., least likely to be spam) [37]. ClueWeb12 provides a larger, more realistic Web setting than other TREC corpora, such as Gov2 Web page collection.

Our experiments mainly aim at measuring and optimizing the end-to-end effectiveness of a multi-stage IR system. As such, we used the TREC 2013 Web Track query set, which provides 50 queries along with relevance assessment of documents from the Clueweb12 dataset. In order to reflect authentic Web usage, the Web Track 2013 queries were developed from the logs and data resources of commercial search engines. In Web Track 2013 there are two types of queries: (1) *faceted topics* and (2) *unfaceted (single-facet) topics* [38]. Faceted topics are more like "head" queries. They are structured as having a representative set of subtopics, with each subtopic corresponding to a popular sub-intent of the main topic. Subtopics make faceted topics less ambiguous as compared to other query sets. Unfaceted (single-facet) topics are intended to be more like "tail" queries with clear questions or intents. Single-facet topic candidates were developed based on queries extracted from search log data that had low-frequency ('tail-like'), but were issued by multiple users. These queries were less than 10 terms in length and had relatively low effectiveness scores across multiple commercial search engines [38]. Thus, the TREC 2013 Web Track query set (referred, henceforth, as TREC 2013) provides queries intended to reflect authentic web usage.

3.3 Training Machine Learning Models

We first obtained the list of documents with relevance judgements. We found 3668 documents in ClueWeb12B with relevance judgements for the TREC 2013 queries. These judged documents are either irrelevant (i.e., relevance label 0) or relevant (i.e., relevance label 1, 2, or 3). To conduct our experiments, we used either the judged documents or the top-K documents (ranked using BM25 [10]) of the TREC 2013 queries. Next, we used the Tesserae framework to extract the features of the documents. After extracting the features, we trained our machine learning models using a 5-fold cross validation strategy. We used three-fifth of the queries (30 queries) for training, one-fifth (10 queries) for validation, and the remaining queries (10 queries) for testing. We have used eight different learning-to-rank (LTR) models for our experiment. Five of them are tree-based: *MART* [6], *LambdaMART* [7], *Obv-LambdaMART* [28], *Obv-MART* [28], *DART* [32], two are linear: *CoordinateAscent* [29], *LineSearch* [30], and lastly, *RankBoost* [31], is a weak learner.

3.4 Metrics

For measuring the efficiency of an IR system, we use runtime as a metric. Alongside, we report the effectiveness of different machine learning algorithms using Normalized Discounted Cumulative Gain of top-10 (NDCG@10), which is a standard metric for measuring effectiveness in IR [39]. Conceptually, NDCG@10 measures the presence of relevant documents in the top-10 list after re-ranked.

$$DCG@10 = \sum_{i=1}^{10} \frac{2^{rel_i} - 1}{log_2(i+1)} \tag{3.1}$$

$$NDCG@10 = \frac{DCG@10}{IDCG@10} \tag{3.2}$$

NDCG@10 (see Equation 3.2) is calculated using the Discounted Cumulative Gain of top-10 (DCG@10) metric as shown in Equation 3.1 and IDCG@10. DCG@10 is the summation of the ratio of a gain function to a discount function for the top-10 positions. The gain function accumulates the relevance labels of the documents in the top-10 list, whereas the discount function discounts the documents present at the bottom of the list. NDCG@10 is eventually computed by taking the ratio of DCG@10 over IDCG@10, which is the DCG@10 of the optimally re-ranked list.

3.5 Testbed

We deploy PISA on a VM with 8 CPUs and 32GB of RAM hosted on a private research cloud. In addition, we utilize an Intel(R) Xeon(R) CPU E7-4850 v4 @ 2.10GHz machine with dual 64 cores and 1 TB of RAM for extracting features under the Tesserae framework. Our Quickrank framework is light; hence, we use it on an Intel(R) Core(TM) i7-8750H @2.20 GHz machine with 16 GB of RAM.

Chapter 4

Experiments

4.1 Experimental Study 1

Research question: What is the correlation between the number of relevant documents (i.e., *Recall*) and the size of the candidate set (i.e., K)?

Study overview: This study only involves the candidate set generation stage (i.e., stage 1) and analyzes the distribution of judged documents of TREC 2013 queries in the top-K documents. It is crucial to study the judged documents because the insights gleaned can be used to justify the trends in other experimental studies. For instance, knowing the recall of the top-K documents can be leveraged to analyze the performance of different machine learning models. As an overview, judged documents contain 2839 documents with relevance label 0 (i.e, irrelevant documents), 636 documents with relevance label 1, 173 documents with relevance label 2, and 20 documents with relevance label 3. For all our experiments, we consider the unjudged documents in top-K as irrelevant based on the TREC 2013 guidelines [38].

Objectives: This study aims to:

1. Find out how the number of relevant documents changes as K decreases from 1000 to 10.

2. Identify the ratio of judged documents to the total documents in the top-K list as K varies from 1000 to 10.

Results: Figure 4.1 shows the distribution of the judged documents and the top-K documents across different relevance labels. First, we observe that most irrelevant documents (2839 documents) are present in judged documents. Top-1000 documents of TREC 2013 queries have 72.4% (2659 out of 3668) judged documents. Interestingly, the top-1000 documents contain nearly all the relevant documents (i.e., relevance label 1,2,3) that are present in the judged documents. To be more precise, 95.7% of documents with relevance label 1, 95.3% of

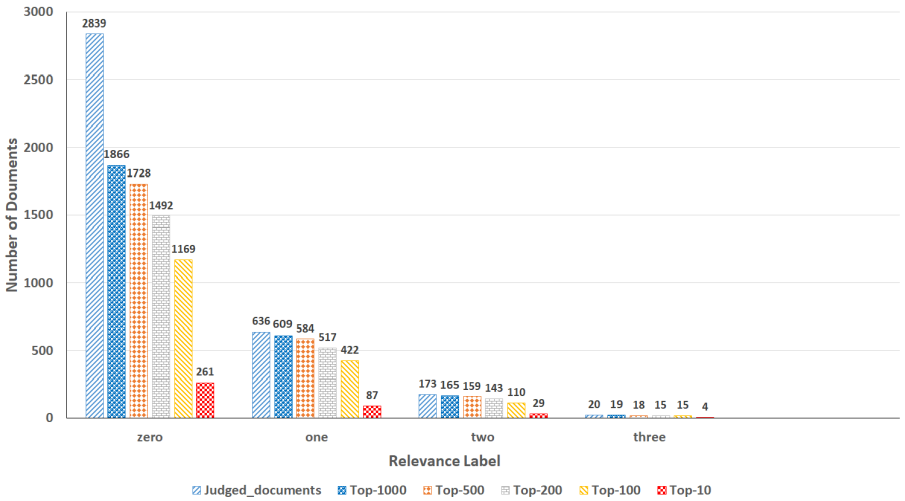


Figure 4.1: Relevance Label vs Number of Documents

documents with relevance label 2, and 95% of documents with relevance label 3 are present in the top-1000 list. This suggests that the top-1000 documents of TREC 2013 queries have an excellent recall.

Second, as the size of the candidate set (K) declines, there is a sharp decrease in the number of judged irrelevant documents. In particular, the number drops from 1866 to 1169 (37.3% drop) as K decreases from 1000 to 100. But, the number of relevant documents decreases comparatively slowly (30.9% drop). This indicates that as K decreases from 1000 to 100, top-K documents get relatively saturated with relevant documents.

Third, when K drops from 100 to 10, there is a significant decrease in the number of relevant documents. The number of documents with relevance label 1 drops from 422 to 87 and, the number of documents with relevance label 2 drops from 110 to 29 (overall 78% drop in relevant documents). This suggests that the top-10 documents have a significantly poor recall as compared to the top-100 documents of the TREC 2013 query set.

Let us now try to analyze the ratio of judged documents to the total number of documents in top-K as K varies from 1000 to 10 (see Figure 4.2). This is critical because the unjudged documents in top-K are assumed to be irrelevant as per the TREC 2013 guidelines [38].

From Figure 4.2, we observe that as K decreases from 1000 to 10, the total number of documents in top-K declines sharply (decreasing by 50% or more). However, the number of judged documents in top-K declines at a slower rate. These two facts combined together indicate that as K drops, the number of unjudged irrelevant documents decreases significantly.

Note that the top-10 documents have the highest concentration of judged documents. But, we know from the first part of this study that the top-10

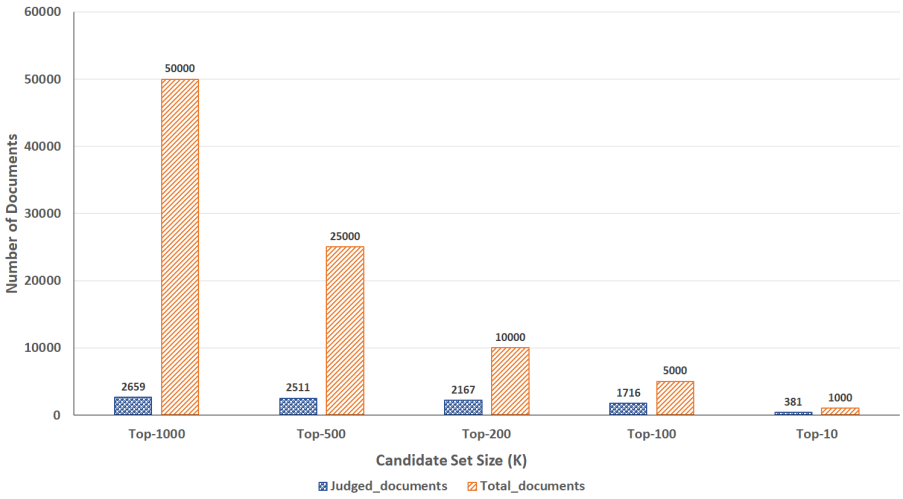


Figure 4.2: Judged Documents vs Total Documents in Top-K

documents have a significantly poor recall compared to the top-100 documents. Hence, the top-100 documents are preferred over the top-10 documents.

Key takeaways:

1. Top-1000 documents of TREC 2013 queries have an excellent recall.
2. As K decreases from 1000 to 10, the number of irrelevant documents (judged and unjudged) decreases significantly in relation to the number of relevant documents.
3. As K drops from 100 to 10, the number of relevant documents decreases significantly (i.e., by 78%), suggesting that the top-10 documents have a considerably poor recall as compared to the top-100 documents.

4.2 Experimental Study 2

Research question: What is the best set of features for machine learning models when trained and tested with judged documents?

Study overview: This study involves the feature extraction stage (i.e., stage 2) and the candidate set re-ranking stage (i.e., stage 3). For this study, we have trained eight different machine learning models with the judged documents of TREC 2013 queries, using seven different sets of features. To begin with, we utilized three core sets of features: (1) *query-dependent* (62 features), (2) *static-document* (13 features), and (3) *pre-retrieval* (155 features). In addition, we considered their combinations to determine which combination will be the best for learning-to-rank (LTR) models.

Objectives: This study aims to:

1. Determine the best set of features for LTR models when trained and tested with judged documents.
2. Identify the best performing machine learning model for re-ranking judged documents.

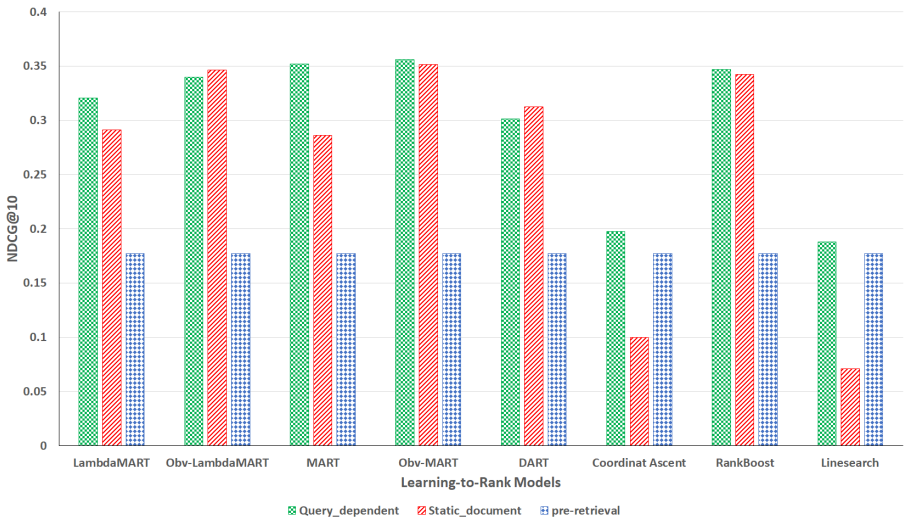


Figure 4.3: Learning-to-Rank Models vs NDCG@10

Results: Figure 4.3 describes the effectiveness (i.e., accuracy) of different LTR models under three core sets of features, using the NDCG@10 metric. We observe that Obv-LambdaMART [28], Obv-MART [28], DART [32], and RankBoost [31] have similar effectiveness on the query-dependent features and the static-document features. In contrast, the remaining LTR models have

higher effectiveness on the query-dependent features as compared to the static-document ones. Overall, this indicates that the query-dependent and static-document features are more effective relative to the pre-retrieval ones.

Analyzing the pre-retrieval features reveals that all LTR models have the same effectiveness (i.e., NDCG@10) value. For a given query, the pre-retrieval features are the same for all of its documents. As such, machine learning models are not able to learn how to re-rank these documents. As an outcome, the re-ranked list produced by the LTR models will be the same as the input list. This justifies the observation that the effectiveness of all LTR models remains the same for the pre-retrieval features.

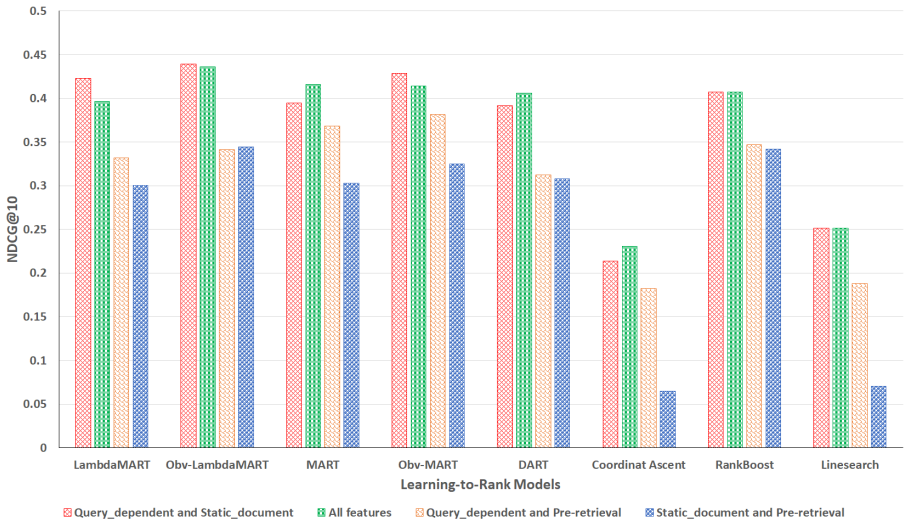


Figure 4.4: Learning-to-Rank Models vs NDCG@10

Figure 4.4 highlights the effectiveness (i.e., accuracy) of LTR models with respect to different combinations of features, using the NDCG@10 metric. It can be observed that for all LTR models, the combination of the query-dependent and static-document features (75 features) results in near best effectiveness. We can further extrapolate that this combination of features has nearly the same effectiveness as the set of all features (230 features). For MART [6], DART [32], and Coordinate Ascent [29], the set of all features slightly outperforms the combination of the query-dependent and static-document features. However, the increase in effectiveness is insignificant. Hence, we recommend the combination of query-dependent and static-document features because using fewer features translates into a gain in efficiency during feature extraction, model training, and model testing.

From Figure 4.4 as well, we conclude that the tree-based models (e.g., LambdaMART, Obv-LambdaMART, etc.,) outperform the linear models (e.g., Coordinate Ascent). Moreover, Obv-LambdaMART or Obv-MART has the

highest effectiveness within each feature class, indicating that the oblivious versions of tree-based algorithms are highly effective. LambdaMART and MART have slightly lower effectiveness (i.e., NDCG@10) values relative to their oblivious counterparts. For the combination of query-dependent and static-document features, DART has similar effectiveness to that of MART and LambdaMART. Furthermore, RankBoost performs similarly to MART and DART.

Lastly, the linear models (i.e., Coordinate Ascent and Linesearch) have the lowest effectiveness under all the feature classes. Across the three core sets of features, linear models perform worst under the static-document features. Moreover, the performance of linear models degrades when the pre-retrieval features are combined with the static-document ones. This indicates that the linear models render inferior with a large number of features (i.e., in high dimensional feature space).

In Figure 4.5, we show the effectiveness of the LTR models with static-document and pre-retrieval features combined versus static-document features only. Clearly, the combination of static-document and pre-retrieval features yields an effectiveness that is akin to the one provided by the static-document features only. In Figure 4.6, we also demonstrate the effectiveness of the LTR models under query-dependent and pre-retrieval features combined versus query-dependent features only. Again, we observe that the combination of query-dependent and pre-retrieval features results in a comparable effectiveness to that of only the query-dependent ones. Therefore, we conclude from Figures 4.5 and 4.6 that combining pre-retrieval features with other core features does not improve the effectiveness of LTR models.

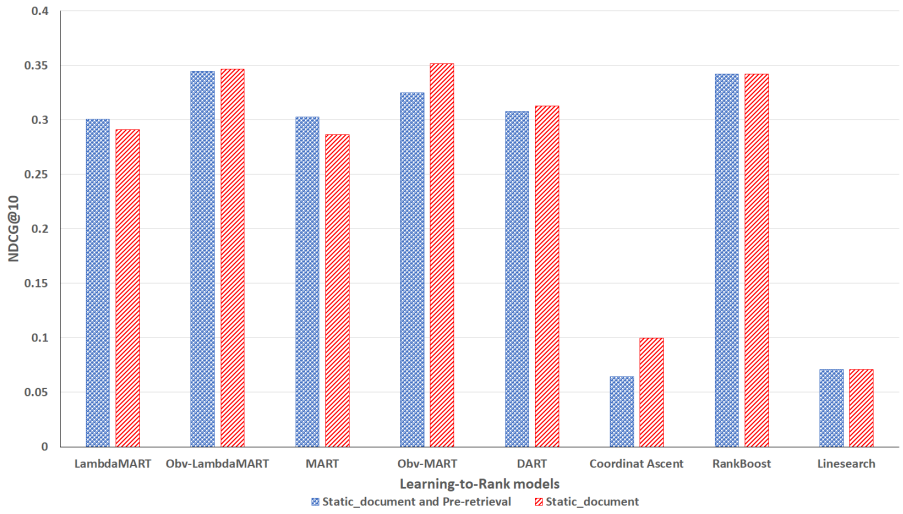


Figure 4.5: Learning-to-Rank Models vs NDCG@10

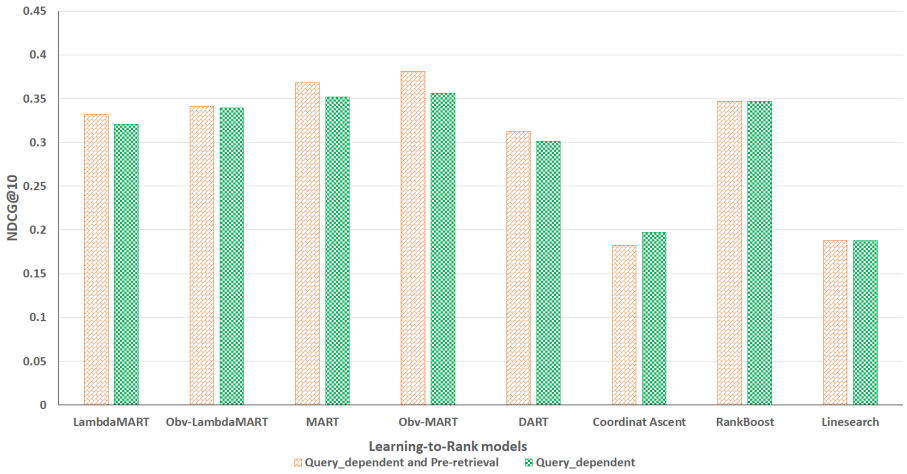


Figure 4.6: Learning-to-Rank Models vs NDCG@10

Key takeaways:

1. Learning-to-rank models show high effectiveness under a combination of query-dependent and static-document features.
2. Obv-LambdaMART or Obv-MART model has the highest effectiveness across all feature classes. Obv-LambdaMART has the highest effectiveness under the query-dependent and static-document feature class.
3. Tree-based models outperform linear models across all feature classes.
4. LTR models are unable to re-rank document lists when trained with pre-retrieval features.
5. The addition of pre-retrieval features to other core features does not improve the effectiveness of LTR models.

4.3 Experimental Study 3

Research question: What is the best size of the candidate set (K) for machine learning models in an end-to-end IR system?

Study overview: This study involves the three stages of a multi-stage retrieval system and aims to find the best size of the candidate set (K) for an end-to-end IR system. In the previous study, it was determined that the combination of query-dependent and static-document features results in near best effectiveness for LTR models trained with judged documents, while in the first study it was revealed that the top-1000 list has an excellent recall. Based on these takeaways, we trained eight different machine learning models with the top-1000 documents, using the combination of query-dependent and static-document features. This study was conducted by simulating a representative end-to-end IR system and it measures the effectiveness produced by different learning-to-rank (LTR) models.

Objectives: This study aims to:

1. Determine the best sizes of the candidate set (K) for different LTR models based on their effectiveness.
2. Find the best performing LTR model for end-to-end IR systems.

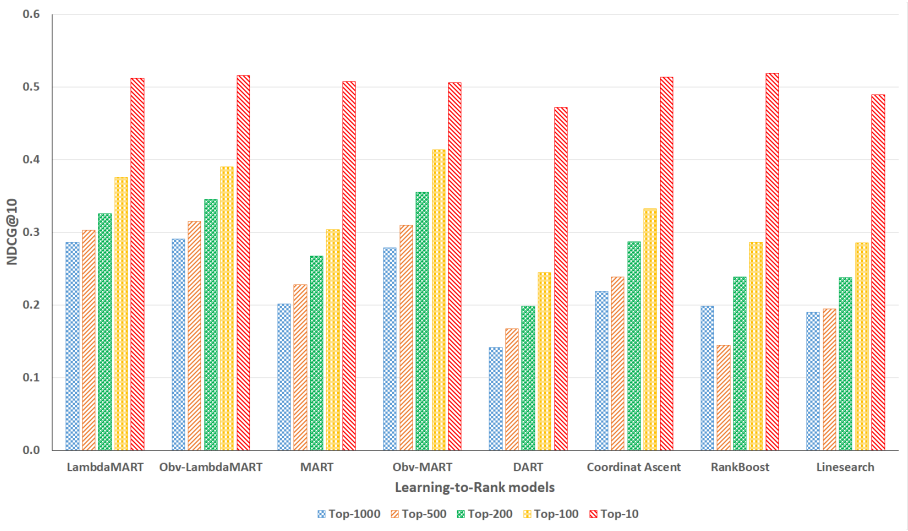


Figure 4.7: Learning-to-Rank Models vs NDCG@10

Results: Figure 4.7 highlights the effectiveness (i.e., accuracy) of LTR models with respect to different candidate set sizes (K), using the NDCG@10 metric. First, we observe that for each LTR model, as K decreases from 1000 to 100,

NDCG@10 (i.e., effectiveness) increases. This behavior can be explained using the conclusions from the first study. In particular, we know that as K decreases from 1000 to 100, the number of irrelevant documents (judged and unjudged) in the top-K decreases sharply compared to the number of judged relevant documents. Moreover, as K drops from 1000 to 100, the ratio of the number of judged documents to the total number of documents in the top-K list increases. These facts suggest that the top-100 documents are more saturated with relevant documents as opposed to the top-1000 documents. Consequently, it becomes less likely for an irrelevant document to find a spot in the top-10 list, when the top-100 documents are re-ranked by a machine learning model. This leads to a conclusion that as the size of the candidate set (K) declines from 1000 to 100, there will be an increase in the effectiveness of the LTR models.

Second, we observe that there is a sharp increase in NDCG@10 as K drops from 100 to 10. This is because of the significantly poor recall of the top-10 documents as compared to the top-100 documents. For this reason, the optimal list produced using the top-10 documents is less effective than the optimal list produced using the top-100 documents. Since NDCG@10 is normalized against the DCG of the optimally re-ranked list (i.e., IDCG@10), a re-ranked list of top-10 documents containing fewer relevant documents receives a higher NDCG@10 score than a re-ranked list of top-100 documents with more relevant documents in the top-10 positions. We believe that this phenomenon is a limitation in the NDCG metric. Given the above, we recommend using the size of the candidate set (K) as 100 because of the poor recall of the top-10 documents.

Now, we elaborate on the effectiveness produced by different machine learning models for K = 100. To begin with, it can be inferred from Figure 4.7 that Obv-MART [28] has the highest effectiveness at K=100, followed by Obv-LambdaMART [28]. This suggests that oblivious versions of tree-based algorithms are highly effective. LambdaMART [7] and MART [6] have slightly lower NDCG@10 values relative to their oblivious counterparts. These observations are consistent with the outcome of the previous study when the LTR models were trained and tested with the judged documents of TREC 2013 queries.

Alongside, DART [32] and RankBoost [31] have lower effectiveness at K=100 as compared to the other tree-based models. However, DART and RankBoost had effectiveness similar to the other tree-based models when they were trained and tested with the judged documents of TREC 2013 queries. This suggests that DART and RankBoost have relatively lower effectiveness when used in end-to-end IR systems.

We also observe that linear models, namely, Coordinate Ascent [29] and Linesearch [30] have NDCG@10 values of 0.33 and 0.28 respectively at k=100. These values are comparable to that of MART and are better than that of DART and RankBoost. However, in the previous study, we concluded that tree-based models outperform linear models when trained and tested with judged documents. This leads to the fact that the nature of the top-K documents and an increase in the training dataset can assist the linear models in enhancing their effectiveness.

Key takeaways:

1. As K decreases from 1000 to 100, NDCG@10 increases for all the machine learning models.
2. As K drops from 100 to 10, there is a sharp increase in NDCG@10 due to the lower recall (fewer relevant documents) of the top-10 documents. As such, we recommend using K = 100.
3. At K = 100, Obv-MART has the highest effectiveness, followed by Obv-LambdaMART, suggesting that these models are the best for end-to-end IR systems.
4. At K = 100, DART and RankBoost have lower effectiveness as compared to other tree-based models.
5. Linear models demonstrate better effectiveness when trained and tested with the top-1000 documents of TREC 2013 queries.

4.4 Experimental Study 4

Research question: How do machine learning models perform on different subsets of query-dependent features?

Study overview: This study involves the feature extraction stage (i.e., stage 2) and the candidate set re-ranking stage (i.e., stage 3). For this study, we have trained eight different machine learning models with the judged documents of TREC 2013 queries, using five different subsets of query-dependent features. These subsets are: (1) *BM25* [10], (2) *Language Modeling with Dirichlet Smoothing (LMDIR)* [11], (3) combination of *Term Frequency and Inverse Document Frequency (TFIDF)* and *probability scores (PROB)* [40], (4) combination of *bose einstein (BE)* [41] and *DPH-DFR* [8], and (5) *Stream* features for different fields of the documents. BM25, LMDIR, and TFIDF are well-known ranking functions used in the candidate set generation stage. In addition, BE and DPH-DFR are probabilistic term-weighting models used regularly in IR. In this study we analyze the performance of learning-to-rank (LTR) models on the commonly used ranking functions and term-weighting techniques.

Objectives: This study aims to:

1. Analyze the performance of different machine learning models on subsets of query-dependent features.

2. Compare the performance of machine learning models on subsets of query-dependent features versus the entire set of query-dependent features.

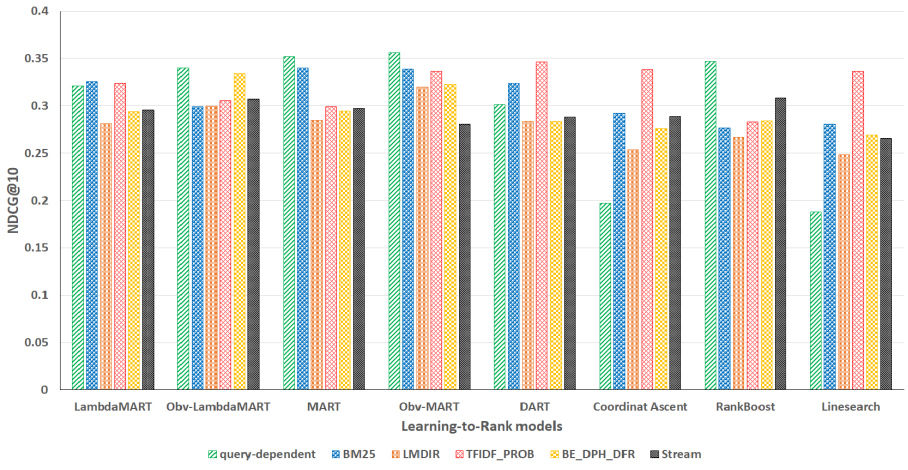


Figure 4.8: Learning-to-Rank Models vs NDCG@10

Results: Figure 4.8 highlights the effectiveness (i.e., accuracy) of LTR models under different subsets of query-dependent features, using the NDCG@10 metric. We observe that for all LTR models except Obv-LambdaMART [28], the BM25 features provide better effectiveness as compared to the LMDIR ones. For Obv-LambdaMART, the BM25 and LMDIR features offer comparable effectiveness. LTR models do not have consistent trends under the remaining subsets of query-dependent features.

We further observe that for tree-based models excluding DART [32], the entire set of query-dependent features outperforms any subsets of query-dependent features. Moreover, Obv-MART has the highest effectiveness among all the LTR models under the entire set of query-dependent features. On the flip side, if only subsets of query-dependent features are considered, DART demonstrates the highest effectiveness under a combination of TFIDF and PROB features.

Next, we analyze the performance of linear models on different subsets of query-dependent features. Both, Coordinate Ascent [29] and Linesearch [30] perform the best under the TFIDF and PROB features combined together. This indicates that judged documents of TREC 2013 queries are more linearly separable with the combination of TFIDF and PROB features relative to other subsets of query-dependent features. Linear models, however, perform the worst under the entire set of query-dependent features. This leads us to conclude that judged documents become less linearly separable as the number and types of features in the feature space are increased. This is consistent with the observation we had in study 2, where the addition of pre-retrieval features to static-document features resulted in degrading the performance of linear models.

Key takeaways:

1. LTR models demonstrate higher effectiveness under the BM25 features as compared to the LMDIR features.

2. For tree-based models, the entire set of query dependent features provides better effectiveness than subsets of query dependent features.

3. Linear models have the best effectiveness on the combined set of TFIDF and PROB features. Conversely, they illustrate the worst effectiveness on the entire set of query-dependent features.

Chapter 5

Recommendations

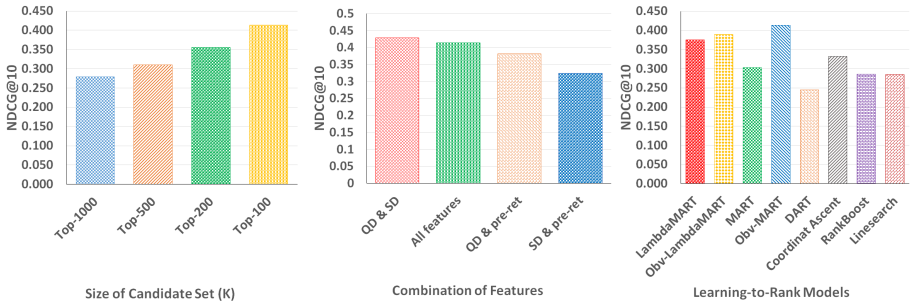
In the previous section, we conducted comprehensive experimental studies that explained the impact of various configuration parameters on the effectiveness (i.e., accuracy) of an IR system. Based on the key takeaways from these experimental studies, we recommend the most favorable configuration parameters and learning-to-rank (LTR) models for maximizing the end-to-end effectiveness of multi-stage retrieval systems. These recommendations are summarized in Table 5.1.

Stage	Recommendation	Justification
Candidate set generation	K = 100	As demonstrated in study 3, this value of K produced the highest ratio of relevant documents to the total number of documents retrieved, thus improving both, recall and precision (see Figure 5.1 (a)).
Feature extraction	Query-dependent and static-document features	As demonstrated in study 2, this combination of features produced comparable effectiveness to the set of all features, thereby providing efficiency gains during feature extraction, training, and testing (see Figure 5.1 (b)).
Candidate set re-ranking	Obv-MART or Obv-LambdaMART	As demonstrated in studies 2 and 3, these models consistently outperformed all other models (see Figure 5.1 (c)).

Table 5.1: Recommendations for Multi-stage Retrieval Systems

In the candidate set generation stage (i.e., stage 1), a retrieval strategy (e.g., WAND [2], MaxScore [3], and LazyBM [4]) and a ranking model (e.g., BM25 [10]) are executed on top of an *inverted index* to retrieve the top-K ranked documents. Based on that we identify three key configuration parameters in this stage: (1) retrieval strategy, (2) ranking model, and (3) size of the candidate set (K). To elaborate, the retrieval strategy traverses over the *posting lists* of the query terms to determine the top-K documents. This process is data intensive and time consuming. As such, various retrieval strategies attempt to safely prune the traversal space in order to expedite the process and boost the efficiency of the candidate set generation stage. Since the efficiency is beyond the

scope of this thesis, we did not consider the retrieval strategy as a configuration parameter in our experiments. On the other hand, the ranking model determines the scores of the documents in the posting lists, thus impacting recall. Note that the end-to-end effectiveness of a multi-stage retrieval system heavily depends on the recall of the top-K documents [1, 5, 42]. This is because the learning-to-rank model deployed in stage 3, capitalizes on the recall of the top-K documents to produce a high precision document list [1]. To this end, it becomes crucial to select a ranking model that yields high recall. For conducting our experiments, we employed the BM25 ranking model to retrieve the top-K documents of the TREC 2013 queries. In experimental study 4.1, we observed that the top-1000 documents have indeed an excellent recall, making BM25 ideal for our studies. Moreover, we concluded that the top-100 documents have a significantly higher recall as compared to the top-10 documents, while improving precision versus top-1000 documents, hence, the choice of using K = 100 as discussed next.



(a) (b) (c)

Figure 5.1: (a) Demonstrates the effectiveness of Obv-MART using different values of K, (b) displays the effectiveness of Obv-MART under different combinations of features (QD stands for query-dependent, SD for static-document, and pre-ret for pre-retrieval), and (c) highlights the performance of LTR models at K = 100.

In experimental study 4.1, we observed that as K decreases from 1000 to 100, the number of irrelevant documents (judged and unjudged) in the top-K list decreases tremendously compared to the number of relevant documents. Furthermore, as K drops from 1000 to 100, the ratio of the number of judged documents to the total number of documents in the top-K list increases. In experimental study 4.3, we exploited these results to explain the increase in the performance of the learning-to-rank (LTR) models at K = 100. We also justified the sharp increase in NDCG@10 at K = 10 using the lower recall of the top-10 documents and the nature of the NDCG metric. Consequently, we recommend using K = 100 for end-to-end multi-stage retrieval systems.

The second stage is the feature extraction stage (i.e., stage 2). The key configuration parameter in this stage is the types of features extracted. Experimental study 4.2 determined the best set of features for the LTR

models via comparing and contrasting their effectiveness under seven different combinations of features. The study found that the LTR models produce near best effectiveness under the combination of query-dependent and static-document features (75 features). In addition, this combination of features demonstrates nearly the same effectiveness as the set of all features (230 features). Subsequently, we recommend this combination for multi-stage retrieval systems because using fewer features translates into a gain in efficiency during feature extraction, model training, and model testing.

The final stage is the candidate set re-ranking stage (i.e., stage 3). The key configuration parameter in this stage is the LTR model. In experimental study 4.2, we observed that Obv-LambdaMART or Obv-MART [28] has the highest effectiveness across all sets of features. This result was verified in experimental study 4.3 as well, where we discovered that Obv-MART has the highest effectiveness at K = 100, followed by Obv-LambdaMART. Therefore, we recommend using Obv-LambdaMART and/or Obv-MART for real-world end-to-end multi-stage retrieval systems.

Chapter 6

Conclusions & Future Work

Modern Information Retrieval (IR) systems involve three stages to achieve desirable effectiveness (i.e., accuracy) and efficiency (i.e., speed). Throughout the last decade, a significant effort was exerted to increase the effectiveness of the candidate set re-ranking stage (i.e., stage 3) in IR systems. For instance, many state-of-the-art LTR models (e.g., Obv-MART [28]) and different types of features (e.g., DPH-DFR [8]) were proposed to enhance the effectiveness of stage 3. During our investigation, we discovered that these LTR models were developed in obliviousness to the configuration parameters of the first two stages of IR systems, namely, the candidate set generation stage (i.e., stage 1) and the feature extraction stage (i.e., stage 2). More precisely, we found that the size of the candidate set (K) and the recall of the top-K documents impact the performance of the LTR models. In addition, we realized that the types of features extracted in stage 2 significantly influence the effectiveness of these LTR models.

To this end, we conjectured that optimizing configuration parameters in stage 1 or stage 2 without considering the impact on stage 3 may not yield optimal end-to-end effectiveness. As such, we studied stages 1, 2, and 3 independently (or *vertically*) and dependently (or *horizontally*), using a comprehensive experimental approach. Specifically, we studied stage 1 vertically (i.e., in-depth) to analyze the recall and nature of top-K documents. Afterwards, we leveraged the insights gained from this vertical analysis to go horizontally and increase the effectiveness of LTR models and, subsequently, IR systems.

As an outcome, we found that: (1) the candidate set size (K) of 100 in stage 1, (2) the combination of query-dependent and static document features in stage 2, and (3) the Obv-LambdaMART or Obv-MART [28] model in stage 3 can collectively produce the best effectiveness for real-world end-to-end multi-stage IR systems. We also identified a surprising limitation in the NDCG@10 metric. Consequently, we proposed a promising research direction for improving NDCG@10 when evaluating document lists with low recalls.

In this thesis, we focused on improving the *effectiveness* of IR systems through a comprehensive experimental strategy. As a future direction, we

propose pursuing a similar approach, but for increasing the *efficiency* of IR systems. This may lead to combining deep effectiveness and efficiency insights in a magnificent way, elevating thereby IR systems to a potentially new level.

References

[1] Nicola Tonellotto, Craig Macdonald, Iadh Ounis, et al. Efficient query processing for scalable web search. *Foundations and Trends® in Information Retrieval*, 12(4-5):319–500, 2018.

[2] Matthias Petri, J. Shane Culpepper, and Alistair Moffat. Exploring the magic of wand. In *Proceedings of the 18th Australasian Document Computing Symposium*, ADCS '13, page 58–65, New York, NY, USA, 2013. Association for Computing Machinery.

[3] Howard Turtle and James Flood. Query evaluation: Strategies and optimizations. *Inf. Process. Manage.*, 31(6):831–850, November 1995.

[4] Mohammad Hammoud Omar Khattab and Tamer Elsayed. Best of two worlds: Faster and more robust top-k retrieval. 2020.

[5] J. Shane Culpepper, Charles L. A. Clarke, and Jimmy Lin. Dynamic cutoff prediction in multi-stage retrieval systems. In *Proceedings of the 21st Australasian Document Computing Symposium*, ADCS '16, page 17–24, New York, NY, USA, 2016. Association for Computing Machinery.

[6] Jerome Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29, 11 2000.

[7] Qiang Wu, Christopher J. Burges, Krysta M. Svore, and Jianfeng Gao. Adapting boosting for information retrieval measures. *Inf. Retr.*, 13(3):254–270, June 2010.

[8] Giambattista Amati, Giuseppe Amodeo, Marco Bianchi, Carlo Gaibisso, and Giorgio Gambosi. Fub, iasi-cnr and university of tor vergata at trec 2008 blog track. 01 2008.

[9] Craig Macdonald, Rodrygo L.T. Santos, Iadh Ounis, and Ben He. About learning models with multiple query-dependent features. *ACM Trans. Inf. Syst.*, 31(3), August 2013.

[10] Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389, April 2009.

[11] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, page 334–342, New York, NY, USA, 2001. Association for Computing Machinery.

[12] Nima Asadi and Jimmy Lin. Effectiveness/efficiency tradeoffs for candidate generation in multi-stage retrieval architectures. In *Proceedings of the 36th*

International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '13, page 997–1000, New York, NY, USA, 2013. Association for Computing Machinery.

[13] Gabriele Capannini, Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, and Nicola Tonellotto. Quality versus efficiency in document scoring with learning-to-rank models. *Inf. Process. Manage.*, 52(6):1161–1177, November 2016.

[14] B. Barla Cambazoglu, Hugo Zaragoza, Olivier Chapelle, Jiang Chen, Ciya Liao, Zhaohui Zheng, and Jon Degenhardt. Early exit optimizations for additive machine learned ranking systems. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, page 411–420, New York, NY, USA, 2010. Association for Computing Machinery.

[15] Tao Qin and Tie-Yan Liu. Introducing letor 4.0 datasets. *arXiv preprint arXiv:1306.2597*, 2013.

[16] Lidan Wang, Jimmy Lin, and Donald Metzler. A cascade ranking model for efficient ranked retrieval. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, page 105–114, New York, NY, USA, 2011. Association for Computing Machinery.

[17] Antonio Mallia, Michal Siedlaczek, Joel Mackenzie, and Torsten Suel. PISA: performant indexes and search for academia. In *Proceedings of the Open-Source IR Replicability Challenge co-located with 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, OSIRRC@SIGIR 2019, Paris, France, July 25, 2019.*, pages 50–56, 2019.

[18] Alistair Moffat and Lang Stuiver. Binary interpolative coding for effective index compression. *Information Retrieval*, 3(1):25–47, 2000.

[19] Sebastiano Vigna. Quasi-succinct indices. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 83–92. ACM, 2013.

[20] Jeff Plaisance, Nathan Kurz, and Daniel Lemire. Vectorized vbyte decoding. *arXiv preprint arXiv:1503.07387*, 2015.

[21] Hao Yan, Shuai Ding, and Torsten Suel. Inverted index compression and query processing with optimized document ordering. In *Proceedings of the 18th international conference on World wide web*, pages 401–410. ACM, 2009.

[22] Andrew Trotman. Compression, simd, and postings lists. In *Proceedings of the 2014 Australasian Document Computing Symposium*, page 50. ACM, 2014.

[23] Daniel Lemire and Leonid Boytsov. Decoding billions of integers per second through vectorization. *Software: Practice and Experience*, 45(1):1–29, 2015.

[24] Shuai Ding and Torsten Suel. Faster top-k document retrieval using block-max indexes. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 993–1002. ACM, 2011.

[25] Antonio Mallia, Giuseppe Ottaviano, Elia Porciani, Nicola Tonellotto, and Rossano Venturini. Faster blockmax wand with variable-sized blocks. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 625–634. ACM, 2017.

[26] L Gallagher et al. Tessearae framework, 2016.

[27] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. Luandri: A clean lua interface to the indri search engine. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, page 1221–1223, New York, NY, USA, 2017. Association for Computing Machinery.

[28] Ilya Segalovich. Machine learning in search quality at yandex. *Invited Talk, SIGIR*, 125, 2010.

[29] Donald Metzler and W Bruce Croft. Linear feature-based models for information retrieval. *Information Retrieval*, 10(3):257–274, 2007.

[30] David G Luenberger, Yinyu Ye, et al. *Linear and nonlinear programming*, volume 2. Springer, 1984.

[31] Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of machine learning research*, 4(Nov):933–969, 2003.

[32] Korlakai Vinayak Rashmi and Ran Gilad-Bachrach. Dart: Dropouts meet multiple additive regression trees. In *AISTATS*, pages 489–497, 2015.

[33] Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Fabrizio Silvestri, and Salvatore Trani. Post-learning optimization of tree ensembles for efficient ranking. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 949–952. ACM, 2016.

[34] Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Fabrizio Silvestri, and Salvatore Trani. X-cle a ver: Learning ranking ensembles by growing and pruning trees. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 9(6):62, 2018.

[35] Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, and Salvatore Trani. X-dart: Blending dropout and pruning for efficient learning to rank. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1077–1080. ACM, 2017.

[36] J Callan et al. The clueweb12 dataset, 2013.

[37] İbrahim Barış Yılmazel and Ahmet Arslan. An intrinsic evaluation of the waterloo spam rankings of the clueweb09 and clueweb12 datasets. *Journal of Information Science*, page 0165551519866551, 2019.

[38] Kevyn Collins-Thompson, Craig Macdonald, Paul Bennett, Fernando Diaz, and Ellen M Voorhees. Trec 2013 web track overview. Technical report, MICHIGAN UNIV ANN ARBOR, 2014.

[39] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, October 2002.

[40] Gerard Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley Longman Publishing Co., Inc., USA, 1989.

[41] Gianni Amati and Cornelis Joost Van Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Trans. Inf. Syst.*, 20(4):357–389, October 2002.

[42] Tie-Yan Liu. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 3(3):225–331, March 2009.