Breaking the Lens of the Telescope
Online Relevance Estimation over Large Retrieval Sets

Rathee, Mandeep; Venktesh, V.; MacAvaney, Sean; Anand, Avishek

DOI
10.1145/3726302.3729910

Publication date
2025

Document Version
Final published version

Published in
SIGIR 2025 - Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval

Citation (APA)
Rathee, M., Venktesh, V., MacAvaney, S., & Anand, A. (2025). Breaking the Lens of the Telescope: Online Relevance Estimation over Large Retrieval Sets. In *SIGIR 2025 - Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 2287-2297). Association for Computing Machinery (ACM). https://doi.org/10.1145/3726302.3729910

Breaking the Lens of the Telescope: Online Relevance Estimation over Large Retrieval Sets

Mandeep Rathee
L3S Research Center
Hannover, Germany
rathee@l3s.de

Venktesh V
Delft University of Technology (TU Delft)
Delft, The Netherlands
v.viswanathan-1@tudelft.nl

Sean MacAvaney
University of Glasgow
Glasgow, United Kingdom
sean.macavaney@glasgow.ac.uk

Avishek Anand
Delft University of Technology (TU Delft)
Delft, The Netherlands
avishek.anand@tudelft.nl

Abstract

Advanced relevance models, such as those that use large language models (LLMs), provide highly accurate relevance estimations. However, their computational costs make them infeasible for processing large document corpora. To address this, retrieval systems often employ a telescoping approach, where computationally efficient but less precise lexical and semantic retrievers filter potential candidates for further ranking. However, this approach heavily depends on the quality of early-stage retrieval, which can potentially exclude relevant documents early in the process. In this work, we propose a novel paradigm for re-ranking called *online relevance estimation* that continuously updates relevance estimates for a query throughout the ranking process. Instead of re-ranking a fixed set of top-k documents in a single step, online relevance estimation iteratively re-scores smaller subsets of the most promising documents while adjusting relevance scores for the remaining pool based on the estimations from the final model using an online bandit-based algorithm. This dynamic process mitigates the recall limitations of telescoping systems by re-prioritizing documents initially deemed less relevant by earlier stages—including those completely excluded by earlier-stage retrievers. We validate our approach on TREC benchmarks under two scenarios: hybrid retrieval and adaptive retrieval. Experimental results demonstrate that our method is sample-efficient and significantly improves recall, highlighting the effectiveness of our online relevance estimation framework for modern search systems.

https://github.com/elixir-research-group/ORE

CCS Concepts

• **Information systems → Retrieval models and ranking**.

Keywords

Relevance Estimation, Hybrid Search, Adaptive Retrieval

SIGIR '25, Padua, Italy.

ACM ISBN 979-8-4007-1592-1/25/07
https://doi.org/10.1145/3726302.3729910

ACM Reference Format:
Mandeep Rathee, Venktesh V, Sean MacAvaney, and Avishek Anand. 2025. Breaking the Lens of the Telescope: Online Relevance Estimation over Large Retrieval Sets. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '25), July 13–18, 2025, Padua, Italy*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3726302.3729910

1 Introduction

Modern search engines are designed around the principle that only a small fraction of documents in a corpus are truly relevant to a given query, many of which can be identified using simple heuristics, such as lexical matching. Telescoping (or cascading) pipelines leverage this property to reduce the number of documents that need to be provided to more accurate (but more computationally expensive) relevance models such as those that use large language models (LLMs) [25, 35, 36, 41, 42]. While this approach usually ensures that highly relevant documents appear at high ranks in the final result, the performance is ultimately limited by the recall of the early-stage retrievers.

The telescoping approach typically employs cost-effective retrievers such as those that rely on lexical [13, 40] or semantic [18, 23] signals and efficient algorithms (such as BlocMaxWAND [12] or HNSW [30]) to perform initial candidate selection. To help ensure high recall, these systems are often combined into hybrid lexical-semantic ensembles [3], or extended using the nearest neighbors of the top documents with adaptive methods [28]. These approaches achieve recall by ensuring broad coverage of potentially relevant documents. Subsequently, machine-learned rankers refine the top-k retrieved documents, optimizing precision-based measures with finer-grained relevance estimates.

Two major shortcomings limit telescoping pipelines. First, recall is inherently constrained by the quality of the initial retrieval stage, leading to the bounded-recall problem. Documents missed during this stage are irretrievably excluded from subsequent ranking, regardless of their relevance to the query. This over-reliance on early-stage retrievers undermines the system's ability to recover highly relevant documents. For example, Figure 1 shows that relevant documents can be present beyond the top-k fold imposed in typical telescoping settings. Second, documents from the early-stage retriever are processed in the order of their initial ranking scores, thereby filtering out documents that do not meet the re-ranking depth. Although the initial ranking may be a good initial

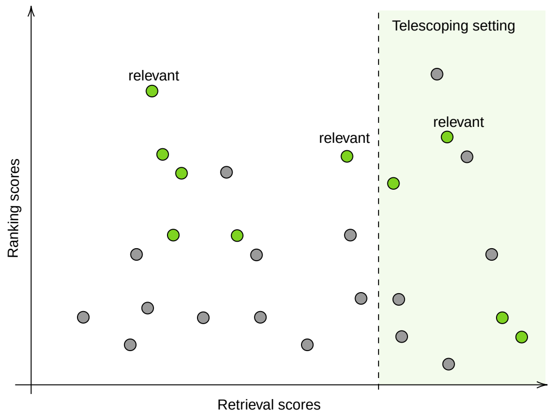


Figure 1: The distribution of retrieval and ranking scores of the retrieved documents. The green region represents the documents selected in the telescoping for ranking. The green documents are selected on the basis of online relevance estimation. The ground truth documents are explicitly labelled as "relevant".

prioritization of documents, we argue that processing the initial ranker's results order becomes suboptimal once the re-ranking model provides higher-quality relevance estimations. Although recent works [19, 20, 28, 37] have proposed adaptive retrieval to overcome the first problem, they still suffer from the second by relying on heuristics for prioritizing the candidate documents.

This work proposes a novel departure from the classical telescoping framework to address these limitations. Our approach, which we call *online relevance estimation* (ORE), introduces a dynamic re-ranking paradigm that iteratively updates relevance estimates for the entire candidate pool throughout the ranking process. Instead of re-ranking a fixed top-k set of documents in one step, our method employs an iterative process that ranks smaller, high-potential subsets. The relevance scores of remaining documents are continuously refined based on the ranking outcomes, enabling previously overlooked documents to be revisited and reconsidered. This approach leverages an online bandit algorithm to optimize relevance estimation dynamically. Figure 2 shows an overview of this process.

We validate our framework on TREC Deep Learning benchmarks under two practical retrieval scenarios: hybrid retrieval and adaptive retrieval. In hybrid retrieval, lexical and dense retrieval methods are fused to generate initial candidates, which are then re-ranked using cross-encoders. We demonstrate that online relevance estimation significantly improves recall by iteratively refining the rankings of a larger pool of documents. In the adaptive retrieval setting, which involves iterative ranking based on neighborhood exploration within a corpus graph, we show that our method surpasses existing approaches by explicitly estimating and updating candidate relevance scores. Unlike current adaptive retrieval methods, which focus on retrieving additional candidates, our approach integrates relevance estimation into the iterative process.

Experimental results highlight that our method is sample-efficient, offering **2**× speedups over state-of-the-art, with the ORE component taking **10**× less time than expensive ranker calls. It also

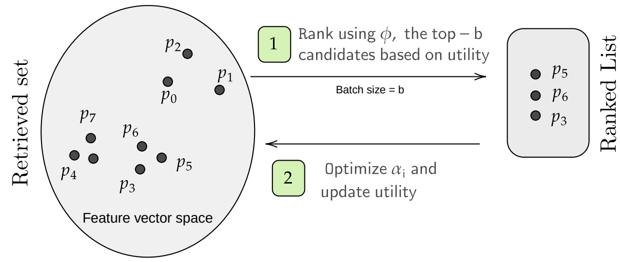


Figure 2: Schematic figure of the Online relevance estimation.

achieves substantial recall improvements, with upto **30.55 %** gains on DL21 for adaptive retrieval and upto **14.12%** gains on DL19 for hybrid retrieval with respect to the corresponding state-of-the-art. With respect to the standard telescoping baseline (BM25»ranker), we achieve improvements of up to **58.53%** on DL22. By bridging the retrieval and ranking stages, our online relevance estimation framework offers a scalable and effective solution to enhance the performance of search systems.

2 Related Work

Recent advancements in document ranking have increasingly relied on complex rankers based on transformer models and, more recently, instruction-tuned models. They are highly effective in providing precise relevance estimates, but computationally expensive. We contextualize our work into hybrid retrieval, adaptive retrieval, and other related ideas on online adaptation for rankings.

2.1 Hybrid Retrieval and Telescoping

Retrieval functions such as BM25 are generally designed to provide fast but less precise relevance estimates. In contrast, complex rankers, including cross-encoders and LLMs, offer far more accurate relevance assessments at the expense of significant computational resources. Due to this tradeoff, complex rankers are typically applied as final-stage ranking functions in a telescoping framework (also referred to as *cascading* or *multi-stage* ranking) [31]. In this framework, an initial ranking is conducted using computationally inexpensive methods like BM25, and only a subset of top-ranked documents is passed to the final stage, where more expensive machine-learned models calculate the final ranking scores. Consequently telescoping paradigm is widely adopted across a variety of domains where strict latency requirements are paramount: web search, e-commerce and live fact-checking systems. Note that there is no restriction on what can be used as a retriever in the first stage. Historically lexical retrieval or BM25 [13, 40, 43] was mostly used as a retrieval function. In more modern search systems dense retrieval [18, 23], learned sparse [13, 26], and hybrid sparse-dense ensembles are used for first-stage retrieval [3, 6, 7, 44].

However, telescoping suffers from a key limitation when the retrieval scores from the first stage do not accurately reflect the relevance of the documents. Retrieval scores are typically used to first-rank documents, and the top-k documents are selected for re-ranking based on their retrieval scores. Since this selection process is typically conducted in a single step, any failure to capture relevant documents in the top-k results can lead to poor recall, ultimately degrading precision in the final rankings. Furthermore, documents that are not passed to the re-ranking stage remain ranked solely according to their initial retrieval scores, which may not reflect their true relevance.

To improve the quality by improving recall, either higher retrieval depths are considered [3, 21], hybrid retrieval [3, 7, 24] or query expansions techniques are employed [4]. However, the choice of documents to rank is still dependent on the retriever score. Unlike these approaches, we dynamically update the relevance estimates for all retrieved candidates by iteratively ranking smaller batches of documents, resulting in improved recall.

2.2 Adaptive Retrieval

The closest approaches to ours are the recently proposed Adaptive Retrieval (AR) methods introduced by MacAvaney et al. [28]. These methods operate on a corpus graph (constructed during an offline phase), which encodes document-document similarities based on lexical or semantic features. Adaptive retrieval methods alternate between the initially retrieved results and the corpus graph neighborhoods of re-ranked documents to select a batch for re-ranking. These methods are fundamentally based on the Clustering Hypothesis [16], which assumes that relevant documents tend to cluster together in the feature space. In Gar [28], only the neighbors of previously ranked documents are explored. More recently, Quam [37] improved upon Gar by selecting documents based on their degree of relatedness to the re-ranked documents.

In contrast to Gar and Quam, which rely on cross-encoders for ranking, Kulkarni et al. [20] proposed a method that uses bi-encoders to re-rank documents. Their approach selects only seed documents from the initial retrieved results and continues exploring the corpus graph neighborhood until the re-ranking budget is exhausted. While adaptive retrieval methods dynamically schedule documents to the ranker, their alternating strategy is heuristic-driven and sample-inefficient. In contrast, our online relevance estimation framework generalizes and simplifies the adaptive retrieval paradigm, offering significant improvements in sample efficiency.

Partially related to our approach are ideas from online learning to rank [14, 22, 45], which learn the parameters of ranking models from user interaction data. However, our approach differs fundamentally from this line of work. Unlike these methods, we do not rely on direct user feedback or address challenges like prioritizing or debiasing rank-sensitive clicks. Moreover, our framework operates on a significantly smaller feature space, allowing it to scale efficiently to large retrieval sizes compared to learning-to-rank models. Other similar works include Reddy et al. [39] and MacAvaney and Wang [29], which learn a new query representation online during re-ranking. Unlike these methods, we use a bandit-based framework and continually refine query representations, thereby selecting better candidate documents at each inference step.

3 Online Relevance Estimation

Typically, telescoping techniques employed for document ranking (also referred to as cascading or multi-stage ranking) rely solely on initial less-precise retrieval scores θ to schedule documents for re-ranking and hence suffer from low recall and precision as relevant documents may be ignored.

Algorithm 1 Online Relevance Estimation

```
Input: Query q, initial retrieved pool R_0, batch size b, budget c,
  number of batches to score m, features vector x⃗_d for document d
Output: Scored pool R_1
  R_1 ← ∅                                         ▷ Scored results
  D_q ← R_0                          ▷ candidate documents (Arms)
  α⃗_1 ← N(0,1) , t ← 1
  do
      D_q ← EstRel(α⃗_t, x⃗_d)    ∀d ∈ D_q    ▷ Assign EstRel scores
      B ← Select(top b from D_q, subject to c)     ▷ using EstRel
      if |R_1| < m · b then
          B ← Score(B, subject to c)                ▷ e.g., monoT5
          α⃗_{t+1} ← min_α⃗ E(α⃗_t, q, d, x⃗_d) ∀d ∈ B
      else
          B ← LookUp(EstRel scores) ∀d ∈ B
      end if
      R_1 ← R_1 ∪ B                          ▷ Add batch to results
      D_q ← D_q \ B                      ▷ Discard batch from Arms
      t ← t + 1
  while |R_1| < c
```

3.1 Problem Definition

The ORE framework is designed to estimate relevance scores for a large pool of retrieved or candidate documents $\mathcal{D}_q$ for a query q such that the relative error between the estimated relevance scores (EstRel) and the cross-encoder scores (ϕ) is minimized. Specifically, for a query q and a candidate document $d \in \mathcal{D}_q$, the objective is to

$$minimize \quad |\phi(q,d) - EstRel(\vec{\alpha}, \vec{x}_d)|^2 \tag{1}$$

where $\phi(q,d)$ represents the accurate relevance score provided by an expensive cross-encoder or ranker, and $\text{EstRel}(\vec{\alpha}, \vec{x}_d)$ is the estimated relevance score derived using simple document features $\vec{x}_d$ and learnable parameters $\vec{\alpha}$. The framework operates under the constraint of a strict budget m, which limits the number of calls to the expensive ranker (ϕ). This efficiency constraint ensures that only a subset of documents is scored directly using ϕ, while the relevance estimates for the remaining documents are derived from EstRel, which serves as a computationally inexpensive proxy for ϕ. The ORE framework presupposes that the cross-encoder ϕ provides reliable relevance scores, which serve as "ground truth" for the estimation process.[1] By approximating ϕ with simple, well-known relevance factors as characteristics (refer to Table 1), ORE aims to achieve an overall improvement in recall by effectively prioritizing highly relevant documents. This allows the framework to balance accuracy and efficiency, ensuring that the relevance estimates closely approximate ϕ while adhering to the computational constraints imposed by the budget m. As a result, ORE provides a scalable solution for large-scale document ranking tasks, achieving high-quality rankings while maintaining computational efficiency.

3.2 The ORE Framework

The problem of relevance estimation in the ORE framework can be formulated as a top-l arms selection problem in stochastic linear

[1] where ϕ itself is an estimation of the true relevance of the document to the query.

Table 1: Description of different features used in ORE. These features can be divided into two levels, Q2DAff and D2DAff.

Feature	Notation	Taxonomy	Source		Description
			Offline	Online	
x_1	$BM25(q,d)$	Q2DAff		✓	Lexical similarity between query and document.
x_2	$TCT(q,d)$	Q2DAff		✓	Semantic similarity between query and document.
x_3	$RM3(q',d)$	D2DAff		✓	Lexical similarity between expanded query using RM3 and document.
x_4	$BM25(d,d')$	D2DAff	✓		Lexical similarity between pair of documents.
x_5	$TCT(d,d')$	D2DAff	✓		Semantic similarity between pair of documents.
x_6	$\text{Laff}(d,d')$	D2DAff	✓		Learnt affinity or similarity between pair of documents [37].

bandits [5, 17]. In this formulation, the *arms* correspond to candidate documents in the initial large retrieval or candidate document pool $\mathcal{D}_q$, the *features vector* ($\vec{x}_d$) encode the properties of each document (as detailed in Table 1), and the *rewards* represent the actual relevance scores ($\phi(q,d)$) obtained from the expensive ranker. For a given query q and a candidate document d, the estimated relevance score computed by ORE is expressed as:

$$\text{EstRel}(\vec{\alpha}, \vec{x}_d) = \vec{\alpha} \cdot \vec{x}_d^T, \tag{2}$$

where $\vec{\alpha}$ represents the learnable parameters of the relevance estimation function. During training, the estimation error, which measures the discrepancy between the estimated relevance score (EstRel) and the actual relevance score (ϕ), is minimized. The error is defined as:

$$E(\vec{\alpha}; q, d, \vec{x}_d) = \frac{1}{2} \left|\phi(q,d) - \text{EstRel}\,(\vec{\alpha}, \vec{x}_d)\right|^2 \tag{3}$$

While classical Multi-Arm Bandit (MAB) approaches iteratively update reward estimates by pulling arms until convergence, they typically require at least linear time in the number of arms per iteration. This makes them computationally impractical for large-scale document retrieval settings, where the candidate document pool can be vast. Therefore, to ensure scalability, ORE constrains ranker calls (ϕ) within a fixed budget m. The framework performs parameter updates for a limited number of batches during re-ranking, learning the parameters $\vec{\alpha}$ for the relevance estimator. For the remaining batches, the learned parameters $\vec{\alpha}$ are used to estimate relevance scores for candidate documents. These estimated relevance scores are then used to add the candidate documents to the final ranked list, prioritizing based on their estimated relevance.

3.3 Query Processing using ORE

Algorithm 1 provides an overview of the ORE procedure. Let q denote the query, R_0 represent the initial pool of retrieved documents, and R_1 the final re-ranked pool of documents, which is initially empty. Let S be the set of top s documents from R_1 that have been re-ranked so far (initially empty), b the batch size, and c the re-ranking budget. The candidate document pool is denoted as $\mathcal{D}_q$, which is initialized with the results retrieved during the first stage (depending on whether the retrieval setup is Hybrid or Adaptive). For each document $d \in \mathcal{D}_q$, let $\vec{x}_d$ denote its feature vector. Each document $d \in \mathcal{D}_q$ is assigned an estimated relevance score, EstRel, computed using Equation 2 with an initial parameter vector $\vec{\alpha}_1$, which is sampled from a normal distribution ($\vec{\alpha}_1 \sim N(0,1)$). The EstRel score quantifies the utility or perceived importance of a document in $\mathcal{D}_q$.

The ORE procedure begins by selecting a batch B of the top b documents from $\mathcal{D}_q$, based on their EstRel scores. These documents are scored using the expensive ranker ϕ (e.g., MonoT5 [33]), and the re-ranked documents are added to R_1. Following this, ORE updates $\mathcal{D}_q$ by either exploring the neighborhood graph (in Adaptive Retrieval) or expanding the retrieval depth (in Hybrid Retrieval) to include additional candidate documents.

To prioritize documents for ranking, the framework recomputes EstRel scores for all documents in $\mathcal{D}_q$ using Equation 2. A new batch B of the top b documents, based on their updated EstRel scores, is selected for ranking. The selected batch is scored using the expensive ranker ϕ, and the parameters $\vec{\alpha}$ of the relevance estimator are updated by minimizing the estimation error as defined in Equation 3. These updated parameters are then used to recompute EstRel scores for the remaining documents in $\mathcal{D}_q$.

The expensive ranker ϕ is used until the condition $|R_1| < m \cdot b$ is satisfied, where m represents the maximum number of batches that can be scored using ϕ. For subsequent documents, the learned parameters $\vec{\alpha}$ are reused to estimate relevance scores, and batches are selected based on their EstRel scores. These selected documents are then added to R_1. The process of updating EstRel scores and selecting batches continues iteratively until the condition $|R_1| < c$ is met, where c is the re-ranking budget. The intuition behind scoring only a subset of documents lies in approximating the relevance of a candidate document $d \in \mathcal{D}_q$ using a learned combination of its features $\vec{x}_d$. By prioritizing and scoring a limited number of batches with ϕ, the learned parameters $\vec{\alpha}$ enable accurate relevance estimation for the remaining documents. This approach eliminates the need for scoring all documents with the ranker, providing significant efficiency gains while maintaining competitive performance.

4 Estimated Relevance in ORE

Relying only on retrieval scores (Q2DAff) would lead to omission of documents which might be relevant, as shown in Figure 1. However, these documents may have closer proximity to documents already deemed relevant as measured by document-document similarity/affinity (D2DAff). If we compute the affinity of the document with respect to a set of documents, it is termed as D2SETAff.

Hence, the choice of features used in ORE is the cornerstone of quality in online relevance estimation. A summary of features employed in ORE in different setups (hybrid and adaptive) is as shown in Table 1. Apart from Q2DAff scores, we also capture the proximity of the document to a small set of documents already deemed relevant by the expensive ranker. The intuition follows from the explore-exploit paradigm of linear stochastic bandits. In the current setting, our goal is to allow for the balance between prioritizing documents with high retrieval scores (exploitation) or provisioning selection of documents which have closer proximity to highly relevant documents despite it's lower retrieval scores (exploration). Note that, ORE is not limited to only the features in the Table 1. The design of ORE algorithm makes it flexible towards the addition of new features.

For a given query q, let R_0 be the initial retrieved results with lexical (x_1) or semantic (x_2) query-document similarities, Q2DAff. and R_1 be the results after re-ranking. Let G_c be the corpus graph. The corpus graph G_c encodes lexical ($x_4 = BM25(d, d')$) or semantic ($x_5 = TCT(d, d')$) document-document similarities, D2DAff. Let G_a be the learnt affinity graph, proposed in Quam [37], which encodes learnt affinity, Laff scores (x_6).

4.1 Hybrid Retrieval using ORE

Hybrid retrieval usually entails employing multiple lexical (BM25) and dense (TCT) retrievers for a high retrieval depth, followed by rank fusion to merge the retrieved lists. These approaches then usually cap the merged results to a lower retrieval depth, ignoring other potentially relevant documents with lower retrieval scores. However, ORE promotes exploration by constructing a candidate pool of documents from the entire merged list. In the hybrid retrieval setup, our goal is to prioritize not only documents with high retrieval scores (Q2Daff) but also to balance the exploration of documents that are in close proximity to documents (D2DAff) already deemed highly relevant. Hence, we carefully select the features from Table 1 reflective of this philosophy

$$\text{Q2DAff}(q, d) = \alpha_1 * BM25(q, d) + \alpha_2 * TCT(q, d)$$

$\forall d \in \mathcal{D}_q$, where $\alpha_1, \alpha_2 \in \vec{\alpha}$.

For D2DAff features in the hybrid retrieval context, we employ both lexical (RM3 i.e., x_3) and semantic scores ($TCT(d, d')$, i.e., x_5). It is critical to note that these D2DAff scores are employed to compute D2SetAff scores, which measure the proximity of a candidate document to a set of highly relevant documents. These highly relevant documents are selected as top-s documents that have already been scored so far from R_1.

$$\text{D2SetAff} = \alpha_3 * RM3(q', d) + \alpha_4 * \frac{\sum_{d' \in S}(\phi(q, d') * TCT(d, d'))}{|S|}$$

where $\alpha_3, \alpha_4 \in \vec{\alpha}$ and q' is the expanded query by using $RM3$ expansion over top re-ranked documents so far in R_1. Note, we simply look up the score of $\phi(q, d')$ since d' is already re-ranked using ranker ϕ. Mapping this to Equation 2, EstRel is computed using $\vec{\alpha} = [\alpha_1, \alpha_2, \alpha_3, \alpha_4]$ and $\vec{x}_d$=[x_1, x_2, x_3, x_5]. The parameters, $\vec{\alpha}$ are learnt using the mechanism described in Section 3.1 and Algorithm 1.

4.2 Adaptive Retrieval using ORE

We adopt a similar philosophy for document prioritization in the adaptive retrieval setup. However, adaptive retrieval is a bit more involved, as the candidate pool $\mathcal{D}_q$ is not static and expands with the addition of neighbors of top-scored documents. Hence, the relevance estimation for the candidate documents is linear in terms of number of documents (arms). Hence, we draw inspiration from top-l arm selection in linear stochastic bandits like LUCB [17] and GIFA [38] which maintain multiple sets such as : 1) arms with high reward estimates and 2) arms with low reward estimates to balance exploration. However, these approaches still sample actual rewards for one arm from each of these lists rendering them computationally infeasible for a large candidate pool. Hence, we maintain two shortlists which represent 1) documents (arms) with high Q2Daff scores, denoted by $U \subset \mathcal{D}_q$, and 2) documents with high D2SetAff scores, denoted by $V \subset \mathcal{D}_q$. The intuition is that since EstRel primarily depends on balancing between documents with high Q2Daff and documents with high D2SetAff scores maintaining shortlists based on these measures help reduce the expanding candidate space and also reduce the impact of documents with noisy estimates.

In the adaptive setting, $\text{Q2DAff}(q, d) = BM25(q, d)$. Given a document affinity graph G_a, the document-document affinity is given by: $\text{D2DAff}(d, d') = G_a(d, d')$ where $G_a(d, d')$ is the edge weight or edge affinity between the source document d and its neighbor d' in the corresponding graph. Note that, we lookup Q2Daff(q,d) and D2DAff(q,d) and compute EstRel $\forall\, d \in U \cup V$ thereby providing an efficient relevance estimation mechanism. Our goal is to primarily balance the exploitation paradigm with the exploration. The exploitation primarily entails selecting documents that have high affinity to the query. Whereas, the exploration paradigm entails scheduling neighbors that may not have high affinity to the query but are closely related to multiple documents deemed to be highly relevant to the query. To accomplish this, we compute the affinity of the candidate document to the ranked set of documents S, denoted as SetD2DAff and defined as

$$\text{D2SetAff}(d, S) = \frac{\sum_{d' \in S \cap N_d}(\text{D2DAff}(d, d'))}{|S \cap N_d|} \tag{4}$$

where $N_d = \text{Neighbours}(d, G_a)$ is the set of neighbors of document d in the learnt affinity graph G_a. The estimated relevance (EstRel) of the candidate document d to the given query q can be better estimated using an average of the relevance (score given by the ranker ϕ) of documents from S in its neighborhood that are already deemed to be highly relevant to the query. Hence we also include this new feature in adaptive retrieval as it naturally fits into the neighborhood-based retrieval philosophy of this setup.

$$x_7 = \frac{\sum_{d' \in S \cap N_d} \text{Score}(q, d')}{|S \cap N_d|} \tag{5}$$

Hence, the features can be combined in the following form for adaptive retrieval:

$$\alpha_1 * \text{Q2DAff}(q, d) + \alpha_2 * \text{D2SetAff}(q, d) + \alpha_3 * x_7$$
$$\text{Score}(q, d') = \begin{cases} \phi(q, d') + \psi(q, d') \; ; \; if\ d'\ is\ scored\ using\ \phi \\ \text{EstRel}(\vec{\alpha}, \vec{x}_d) \; ; \; otherwise \end{cases} \tag{6}$$

where ψ is a dual encoder[2]. We look up the scores from ϕ, since the documents are already in R_1 ($d' \in S' \subseteq R_1$), where S' is set of top s documents in previous iteration. Mapping this to Equation 2, EstRel is computed using $\vec{\alpha} = [\alpha_1, \alpha_2, \alpha_3]$ and $\vec{x}_d$=[x_1, x_6, x_7].

Note that all the above computations for hybrid or adaptive retrieval setups are vectorized and computed for a batch of documents at a time. We present at the document level for ease of understanding. Also, note that $\mathcal{D}_q$ get updated after scoring each batch B with

[2] We use inexpensive dual encoder, TAS-B [15] for better numerical stability.

the neighbors in G_a of each document $d \in B$, i.e., $\mathcal{D}_q \leftarrow \mathcal{D}_q \cup$ Neighbors(d,G_a), but we maintain shortlists as discussed earlier.

5 Experimental Setup

In this work, we demonstrate the effectiveness of online relevance estimation in two commonly used recall-improving scenarios: *hybrid retrieval* and *adaptive retrieval*. To evaluate our approach, we address the following research questions:

RQ1: How effective is ORE compared to existing approaches for hybrid and adaptive retrieval setups?
RQ2: How helpful is the utility (estimated relevance) in prioritizing documents for retrieval?
RQ3: How efficient is ORE compared to existing approaches for adaptive retrieval?
RQ4: How much time does estimated relevance take compared to expensive ranker calls?

5.1 Datasets and Measures

We perform experiments on the MSMARCO passage corpus [32] (with 8.8 M passages) and validate our approach on the TREC Deep Learning 2019 (DL19) [10] and 2020 (DL20) [11] test sets. The DL19 set has 43 queries with an average of 58.1 relevant documents, and DL20 has 54 queries with an average of 30.9 relevant documents. Further, we use the MSMARCO passage-v2 corpus [2] (with 138.4 M passages) and evaluate on TREC DL21 [8] and DL22 [9] test sets. The DL21 has 53 queries with an average of 50.2 relevant documents and DL22 has 76 queries with an average of 58.4 relevant documents. We use the de-duplicated MSMARCO-passage-v2 corpus and both DL21 and DL22 qrels. We measure the ranking performance by nDCG@c, and retrieval by recall@c at different re-ranking budgets $c \in \{50, 100, 1000\}$. We re-use the BM25, TCT based corpus graphs.

5.2 Retrieval and Ranking Models

We mainly use lexical and semantic first-stage retrievers. For lexical retrieval, we use BM25 [40]. We use a Terrier [34] index of the MSMARCO passage corpus. While for semantic retrieval, we use TCT [23] which is based on the TCT-Colbert model, and use the TCT-ColBERT-HNP[3] model for encoding queries and documents. We retrieve documents based on the budget (in the adaptive retrieval setting) or retrieval depth (in the hybrid retrieval setting).

We use the MonoT5-base model [33] (in short MonoT5, a cross-encoder) as the ranker model which is fine-tuned on the MSMARCO corpus. We also use MonoT5 as a retriever on the MSMARCO passage corpus by scoring all documents exhaustively for a query. Also, we do ablation using the fine-tuned pointwise LLM ranker called RankLLaMA [25], which is built upon LLaMA-2-7B[4] and trained for ranking the top documents from the RepLLaMA retriever.

5.3 Baselines and Implementation

To compare the effectiveness of our proposed method, we use re-ranking, hybrid, and adaptive retrieval baselines. We use a standard telescoping re-ranking baseline, retriever followed by ranker, by re-ranking top retrieved documents based on the re-ranking budget c. We denote this ranking baseline by BM25»MonoT5. We also compare our approach with pseudo-relevance feedback (PRF) based RM3 [1] expansion over documents retrieved using BM25.

Hybrid Retrieval. For hybrid retrieval, we use two, BM25 and TCT, retrievers for the first stage, and retrieve 1000 documents exhaustively. We apply Reciprocal Rank Fusion [7] (RRF) over these two rankings and take the top c (budget) documents based on their reciprocal rank scores. We also use Convex Combination [3, 44] (CC) of scores given by BM25 and TCT retriever with interpolation parameter α is set to 0.5 [5].

Adaptive Retrieval. For adaptive retrieval, we use mainly Gar [28] and Quam [37]. Both Gar and Quam alternate between first-stage results and neighborhood graph and prepare the batch of documents for reranking. For both Gar and Quam, we use BM25 and TCT-based corpus graphs with 16 neighbors. The type of corpus is indicated in subscript, for example, Gar with BM25 based corpus graph is denoted by Gar_{BM25}. We use the official implementation to reproduce these baselines.

5.4 Hyperparameters and Tuning

For our experiments, we use re-ranking budget $c \in \{50, 100, 1000\}$, and batch size is set to 16. We mainly use the corpus graphs with 16 neighbors. We use DL19 set as a validation set for tuning hyperparameters and DL20, DL21, and DL22 as test sets. For *RM*3, we set *fb_docs* to 5 and *fb_terms* to 10, and the *original_query_weight* to 0.3. We set $|S| = 10$ for all budgets in hybrid retrieval. We set $|U| = 35, |V| = 25$ for different re-ranking budgets c. For the adaptive retrieval setup, we set the size of the set S to calculate the D2SetAff depending upon the budget. For budgets c of 50, 100, and 1000, we set $|S|$ to 10, 25, and 150, respectively. All of our experiments are done on NVIDIA H100 GPU with 96 GB of RAM.

6 Experimental Results

6.1 Effectiveness of ORE

In the first experiment, we evaluate the effectiveness of online relevance estimation over the telescoping strategy used over standard, hybrid, and adaptive retrieval. To address **RQ1**, we evaluate ORE on TREC-DL 2019 and 2020 datasets, comparing its performance to state-of-the-art methods in hybrid and adaptive retrieval setups in Tables 2 and 3. Firstly, online relevance estimation outperforms baseline ranking performance in telescoping settings i.e., BM25»MonoT5 (up to **58.53%** on DL22 at budget $c = 100$). Secondly, ORE outperforms the pseudo-relevance feedback baseline (BM25»MonoT5 w/ RM3) at all settings.

6.1.1 Hybrid Retrieval. We now turn our attention to hybrid retrieval. As expected, we confirm that both RRF»MonoT5 and CC» MonoT5 convincingly outperformed BM25»MonoT5 at all retrieval depths. This is because using hybrid retrieval balances the complementary lexical and semantic signals. We find that ORE further improves beyond this baseline, achieving statistically significant performance gains over both RRF and CC. For instance, from Table 3, we observe substantial gains, where ORE outperforms CC

[3] https://huggingface.co/castorini/tct_colbert-v2-hnp-msmarco
[4] https://huggingface.co/meta-llama/Llama-2-7b-hf
[5] As [3] mentioned that the CC methods are sensitive to α, we follow the insight from [44] that $\alpha = 0.5$ works best for lexical and semantic interpolation for the MSMARCO corpus.

Table 2: Effectiveness comparison of ORE with hybrid and adaptive retrieval methods on TREC DL19 and DL20 test sets. Significant improvements using paired t-test, $p < 0.05$, with Bonferroni correction, over CC, RRF, baseline (BM25»MonoT5), Gar, and Quam are marked with B, C, R, G and Q respectively. The best scores are highlighted in bold.

		$c = 50$			$c = 100$			$c = 1000$		
Dataset	Pipeline	nDCG@10	nDCG@c	Recall@c	nDCG@10	nDCG@c	Recall@c	nDCG@10	nDCG@c	Recall@c
DL19	**Exhaustive Retrieval**									
	MonoT5	0.672	0.625	0.512	0.672	0.611	0.599	0.672	0.691	0.834
	Hybrid Retrieval: (BM25 & TCT)									
	RRF»MonoT5 [R]	**0.735**	0.658	0.513	0.729	0.664	0.637	**0.703**	0.740	0.879
	CC»MonoT5 [C]	0.729	0.650	0.489	**0.730**	0.650	0.626	0.698	0.738	0.878
	ORE	0.734	^{RC}**0.683**	^{RC}**0.558**	0.721	^{RC}**0.688**	^{RC}**0.675**	**0.703**	**0.741**	**0.882**
	Adaptive Retrieval									
	BM25»MonoT5 [B]	0.681	0.541	0.389	0.699	0.563	0.488	0.719	0.697	0.755
	w/ RM3	0.682	0.560	0.409	0.686	0.574	0.507	0.728	0.717	0.786
	w/ Gar_{BM25} [G]	0.689	0.565	0.417	0.716	0.594	0.539	0.727	0.742	0.836
	w/ $\text{Quam}_{\text{BM25}}$ [Q]	**0.698**	0.597	0.460	**0.729**	0.639	0.594	**0.742**	**0.770**	**0.874**
	w/ ORE_{BM25}	**0.698**	$^{GQ}_{B}$**0.640**	$^{GQ}_{B}$**0.509**	0.711	$^{G}_{B}$**0.653**	$^{G}_{B}$**0.619**	0.723	$_{B}$0.759	$_{B}$**0.874**
DL20	**Exhaustive Retrieval**									
	MonoT5	0.649	0.592	0.576	0.649	0.593	0.670	0.649	0.682	0.852
	Hybrid Retrieval: (BM25 & TCT)									
	RRF»MonoT5 [R]	**0.721**	0.655	0.633	0.707	0.659	0.725	0.676	0.727	0.885
	CC»MonoT5 [C]	0.718	0.654	0.632	**0.709**	0.660	0.721	**0.681**	0.727	0.884
	ORE	0.720	**0.674**	**0.658**	0.702	^{RC}**0.683**	^{RC}**0.759**	0.676	^{C}**0.731**	^{C}**0.892**
	Adaptive Retrieval									
	BM25»MonoT5 [B]	0.676	0.559	0.478	0.685	0.581	0.584	**0.720**	0.711	0.807
	w/ RM3	0.684	0.585	0.520	0.701	0.611	0.633	**0.720**	0.729	0.834
	w/ Gar_{BM25} [G]	0.690	0.577	0.496	0.703	0.607	0.617	0.714	0.750	0.884
	w/ $\text{Quam}_{\text{BM25}}$ [Q]	**0.714**	0.615	0.553	**0.717**	**0.652**	0.678	0.709	0.756	**0.901**
	w/ ORE_{BM25}	0.684	$^{G}_{B}$**0.621**	$^{G}_{B}$**0.583**	0.681	$^{G}_{B}$0.651	$^{G}_{B}$**0.705**	0.700	**0.757**	$_{B}$0.892

by **11.74 %** and RRF by **17.12%** for Recall@100 on DL21. We also observe that ORE improves Recall@100 on DL22 by **7.46 %**, when compared to CC and by **14.09%** when compared to RRF. Further on DL19, Recall@50 improves from 0.489 to 0.558 (an improvement of **14.11%**) and from 0.513 to 0.558 (an improvement of **8.9%**) in CC and RRF respectively. Similar trends are observed across different retrieval budgets, with ORE delivering consistent gains.

These improvements can be primarily attributed to ORE's online relevance estimation capability, which prioritizes documents dynamically based on the current estimate of the relevance. Unlike fusion-based methods that select the *top-k* merged documents based on a one-shot fusion score and ignore others, ORE captures potentially relevant documents with low initial retrieval scores by re-prioritizing them for scoring based on new ranking evidence. Our Multi-Arm Bandits-based online estimation procedure trades off exploration (scheduling low-ranked documents) with exploitation (scoring top-ranked documents), thereby effectively learning the tradeoffs between relevance factors modelled as features. Given our small feature space and linear classifier, ORE can perform this reprioritization efficiently.

6.1.2 Adaptive Retrieval. We also compare ORE to state-of-the-art adaptive retrieval methods, including Gar and Quam. Unlike the hybrid setting, where the retrieval set is fixed, in the adaptive retrieval setting, we adaptively explore the retrieved document space. Our results indicate that ORE outperforms these approaches across various retrieval budgets, with significant gains at lower budgets. For example, on DL21, we observe that ORE advances Recall@50 to **0.406** providing gains of up to **30.55%** over Quam and up to **22.66%** over Gar. On TREC-DL 2019 Recall@50 increases from 0.460 (Quam) and 0.417 (Gar) to 0.509 (**10.65%** and **22.06%**, respectively). These gains arise from the principled document selection strategy employed by ORE. Existing methods like Gar and Quam alternate between first-stage retrieval results and neighborhood lists. We believe that this alternating strategy was proposed in the spirit of ensuring the robustness of results and might be sometimes less sample efficient. For example, the algorithm is forced to schedule documents from the retrieved list to be ranked even though the retrieval scores are low and indicate low relevance. ORE departs from the alternating scheduling strategy by re-estimating document utility over all the candidate documents – from the retrieved results or the neighborhood graph. This approach enables the balanced exploitation of documents retrieved in the first stage and the exploration of related documents identified in their neighborhoods. As a result, ORE prioritizes relevant documents at each iteration that may have low initial retrieval scores but high estimated utility. Surprisingly, ORE also outperforms the exhaustive retrieval pipeline, which uses an expensive scorer to evaluate all documents in the corpus without first-stage retrieval. This highlights the effectiveness of ORE's utility estimation in reducing noise and focusing on potentially relevant documents.

Insight 1: ORE achieves high recall in both hybrid and adaptive

Table 3: Effectiveness comparison* of ORE with hybrid and adaptive retrieval methods on TREC DL21 and DL22 test sets. The letter in subscript or superscript shows significant improvements (using paired t-test, $p < 0.05$, with Bonferroni correction) over the corresponding baseline.

		c = 50		c = 100	
Dataset	Pipeline	nDCG@c	Recall@c	nDCG@c	Recall@c
DL21	**Hybrid**				
	RRF»MonoT5 [R]	0.576	0.401	0.558	0.520
	CC»MonoT5 [C]	0.584	0.419	0.569	0.545
	ORE	^{R}**0.604**	^{R}**0.444**	^{RC}**0.609**	^{RC}**0.609**
	Adaptive				
	BM25»MonoT5 [B]	0.436	0.242	0.433	0.331
	w/ RM3	0.455	0.274	0.457	0.375
	w/ GAR_{BM25} [G]	0.457	0.290	0.465	0.414
	w/ $\text{QUAM}_{\text{BM25}}$ [Q]	0.478	0.310	**0.499**	0.454
	w/ ORE_{BM25}	$^{GQ}_{B}$**0.503**	$^{GQ}_{B}$**0.364**	$_{B}$0.481	$^{G}_{B}$**0.463**
	w/ GAR_{TCT} [G]	0.502	0.331	**0.520**	0.489
	w/ QUAM_{TCT} [Q]	0.491	0.311	0.518	0.477
	w/ ORE_{TCT}	$^{GQ}_{B}$**0.532**	$^{GQ}_{B}$**0.406**	$_{B}$0.512	$_{B}$**0.502**
DL22	**Hybrid**				
	RRF»MonoT5 [R]	0.452	0.260	0.430	0.341
	CC»MonoT5 [C]	0.459	0.278	0.433	0.362
	ORE	^{RC}**0.481**	^{R}**0.297**	^{RC}**0.459**	^{RC}**0.389**
	Adaptive				
	BM25»MonoT5 [B]	0.290	0.115	0.275	0.164
	w/ RM3	0.287	0.115	0.275	0.161
	w/ GAR_{BM25} [G]	0.287	0.121	0.290	0.191
	w/ $\text{QUAM}_{\text{BM25}}$ [Q]	**0.308**	0.135	**0.303**	**0.196**
	w/ ORE_{BM25}	0.292	**0.137**	0.284	0.195
	w/ GAR_{TCT} [G]	0.329	0.157	**0.348**	0.256
	w/ QUAM_{TCT} [Q]	0.329	0.155	0.334	0.237
	w/ ORE_{TCT}	$^{GQ}_{B}$**0.364**	$^{GQ}_{B}$**0.206**	$_{B}$0.342	$_{B}$**0.260**

*We omit nDCG@10 our focus is more on retrieval. Additionally, prior works find that nDCG@10 value saturates quickly during re-ranking [27].

retrieval settings by dynamically learning to prioritize documents through an inexpensive estimation of relevance scores.

6.2 Significance and Quality of Estimated Utility

While the overall performance demonstrates that the proposed utility estimation aids in prioritizing documents, it does not provide insights into its absolute quality or its ability to serve as a reliable proxy for the expensive ranker's scores. To answer **RQ2**, we evaluate the quality of estimated scores in the hybrid and adaptive retrieval setups.

6.2.1 Hybrid Retrieval. We analyze the error between the estimated relevance (EstRel) and the actual relevance scores from the ranker for various ranker call budgets (m). Here, the budget for ranker calls m represents the number of batches of documents scored by the ranker, with $m \cdot b \leq c$. The error for $c = 100$ and $b = 16$ across $m = 1, \ldots, 7$ is shown in Figure 3b.

At $m = 1$, the error is high because the parameters used for estimating utility are initialized randomly, resulting in poor relevance approximations. However, as m increases, the utility estimates improve significantly. For instance, at $m = 2$, only 32 samples are scored, but the learned parameters enable a sharp reduction in

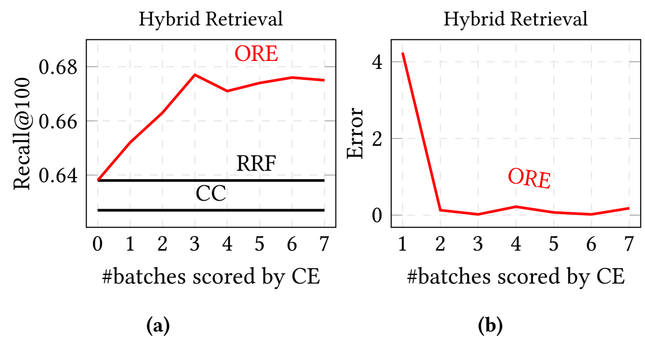


Figure 3: Recall (left) and estimation error (right) comparison for hybrid retrieval setting on the TREC DL19 dataset when the number of batches of scored by cross-encoder (CE) varies for ORE for ranking budget of 100 and batch of size 16.

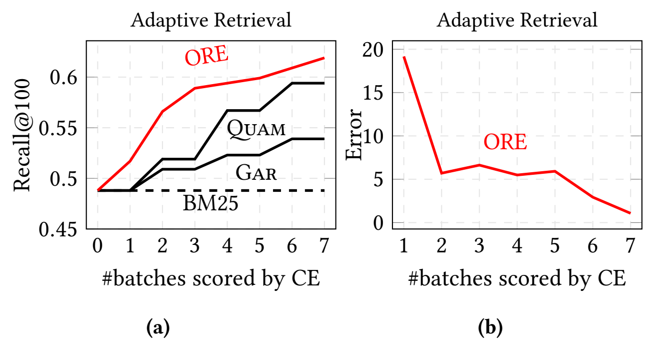


Figure 4: Recall (left) and estimation error (right) comparison on the TREC DL19 dataset for adaptive retrieval, for ranking budget of 100 and batch of size 16.

error, closely approximating the actual relevance scores. As more samples are scored with increasing m, the error continues to decline steadily, reflecting better utility estimation.

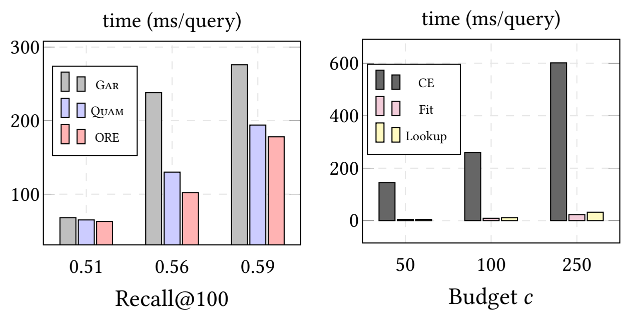
This trend is further supported by Figure 3a, which shows that when only 16 documents are scored, ORE already outperforms traditional hybrid retrieval methods, such as RRF and CC. By estimating high-quality utility scores for the remaining documents, ORE achieves superior performance even with minimal ranker calls.

6.2.2 Adaptive Retrieval. For adaptive retrieval, we analyze the error in estimated utility, as illustrated in Figure 4b. The results reveal a trend similar to the hybrid retrieval setup: the error decreases gradually as the cross-encoder budget (m) increases, with a sharp decline observed at the maximum budget ($m = 7$), where more samples are scored. Additionally, we examine the relationship between the ranker budget and Recall@100 for TREC-DL 2019 (DL19) in Figure 4a. Across all ranker budgets (m), ORE consistently outperforms state-of-the-art adaptive retrieval methods, such as Gar and Quam. Notably, even at $m = 1$, ORE demonstrates superior performance by using its utility estimates as a proxy for ranking documents, avoiding frequent calls to the expensive ranker. This result highlights the high quality of the estimated utility scores, which enable ORE to prioritize relevant documents through principled exploration.

The observed improvement is attributed to the heuristic-based document selection strategies employed by Gar and Quam, which alternate between the initial ranked list and the neighborhood of

Table 4: Mean re-ranking latency per query (in ms) using MonoT5 and RankLLaMA rerankers when the first-stage retrieval of different budgets (c) is done using BM25. The number of batches re-ranked ORE is enclosed in parentheses.

	MonoT5						RankLLaMA					
	time (ms/query)			Recall@c			time (ms/query)			Recall@c		
c	GAR	QUAM	ORE	GAR	QUAM	ORE	GAR	QUAM	ORE	GAR	QUAM	ORE
50	179.92	173.21	125.91(2)	0.417	0.460	0.500	6269.77	6027.12	3925.54(2)	0.421	0.449	0.492
100	356.53	328.82	272.04(4)	0.539	0.594	0.594	12746.55	12074.67	7830.41(4)	0.542	0.600	0.600
250	877.19	816.90	599.24(8)	0.692	0.745	0.715	32312.97	30539.78	16092.16(8)	0.684	0.761	0.719
1000	3418.98	3219.45	1848.26(8)	0.836	0.874	0.827	127617.45	120885.39	16327.25(8)	0.854	0.881	0.829
			2188.29(16)			0.841			31939.78(16)			0.853



(a) **(b)**

Figure 5: Computational efficiency of our proposed method ORE in comparison to adaptive retrieval approaches (left) and overheads from different components (right) in ORE.

scored documents. At lower ranker budgets, these methods score only a limited number of documents and backfill the remaining slots with scores from the initial retrieval results. In contrast, ORE employs a learned utility estimator that performs principled exploration. It dynamically prioritizes documents from the initial retrieval results and their neighborhoods in each batch until the budget c is fully utilized.

Insight 2: Online relevance/utility estimation of ORE works well across hybrid and adaptive retrieval settings and closely approximates actual relevance estimates from the ranker.

6.3 Computational Efficiency of ORE

To address **RQ3**, we demonstrate the latency and sample efficiency gains provided by ORE over GAR and QUAM in the adaptive retrieval setting. In Figure 5a, we present the time taken by ORE and contemporary adaptive retrieval methods to achieve similar recall performance. Specifically, to reach a Recall@100 of 0.56, ORE requires only 2 cross-encoder calls (*102 ms/query*), providing a speedup of **2**× compared to GAR, which takes 8 calls (*238 ms/query*). This highlights ORE's ability to achieve higher recall with fewer scored samples due to its efficient online relevance estimation.

From Table 4, we observe that ORE consistently outperforms existing adaptive retrieval methods in terms of both latency and Recall@c when using an expensive ranker such as RankLLaMa. On average, ORE delivers speedups of **2**×–**3**× and, in certain scenarios, achieves up to **9**× speedups for $c = 1000$ across different budgets compared to GAR and QUAM. These improvements primarily stem from the sample-efficient nature of ORE, which requires fewer scored samples to estimate utility scores for the remaining documents. These utility scores serve as reliable proxies for actual relevance scores, significantly reducing the costly ranker calls.

Further, to answer **RQ4**, we provide a breakdown of the time taken by individual components of ORE in Figure 5b for $c = \{50, 100, 250\}$, batch size $b = 16$ on DL19. These components are, namely, the expensive ranker calls (denoted by CE), feature construction (denoted by Lookup), and parameter updates (i.e., fitting of $\vec{\alpha}$ parameters, denoted by Fit). As we discussed earlier, during re-ranking, the expensive ranker contributes the most in the latency overhead. We observe similar insights here. For example, at budget $c = 250$, the total time for re-ranking is around 657 ms/query, out of which the cross encoder (CE) contributes around 92% (602 ms/query) of the time. The feature lookup takes only 32.2 ms/query (18× less time) compared to 602 ms/query for ranker calls. Similarly, the time taken to learn and update α parameters takes only 22.9 ms/query. Hence, the core component for relevance estimation (parameter fitting and feature lookup) takes **10** × less time than ranker calls at the per query level.

Insight 3: ORE is sample efficient when compared to state-of-the-art adaptive retrieval methods. It requires fewer documents scored by the expensive ranker on average. It provides speedups of upto **2**× for standard rankers like MonoT5 and upto **9** × for more expensive LLM-based rankers like RankLLaMa.

We further explore the importance of different features in ORE, in the Appendix released with our GitHub repository.

7 Conclusion

In this work, we introduce a novel paradigm of dynamically ranking retrieved documents by using online relevance estimation. We propose a departure from the progressive filtering approach popularized by the telescoping method that only ranks documents with high retrieval scores, ignoring other retrieved documents. Instead, we propose to dynamically keep relevance estimates for every retrieved document based on a small set of features based on well-known relevance factors. These estimates are refined dynamically by incorporating ranking scores encountered during the ranking process. Our experiments over four TREC-DL test sets in the hybrid and adaptive retrieval settings clearly show that basic instantiations of online relevance estimation are quite effective and outperform other telescoping and adaptive retrieval baselines.

Acknowledgments

This work is partially supported by the German Research Foundation (DFG), under the Project IREM with grant No. AN 996/1-1.

References

[1] Nasreen Abdul-Jaleel, James Allan, W Bruce Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Mark D Smucker, and Courtney Wade. 2004. UMass at TREC 2004: Novelty and HARD. *Computer Science Department Faculty Publication Series* (2004), 189.

[2] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2018. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. arXiv:1611.09268 [cs.CL] https://arxiv.org/abs/1611.09268

[3] Sebastian Bruch, Siyu Gai, and Amir Ingber. 2023. An analysis of fusion functions for hybrid retrieval. *ACM Transactions on Information Systems* 42, 1 (2023), 1–35.

[4] Claudio Carpineto and Giovanni Romano. 2012. A survey of automatic query expansion in information retrieval. *Acm Computing Surveys (CSUR)* 44, 1 (2012), 1–50.

[5] Arghya Roy Chaudhuri and Shivaram Kalyanakrishnan. 2019. PAC identification of many good arms in stochastic multi-armed bandits. In *International Conference on Machine Learning*. PMLR, 991–1000.

[6] Tao Chen, Mingyang Zhang, Jing Lu, Michael Bendersky, and Marc Najork. 2022. Out-of-domain semantics to the rescue! zero-shot hybrid retrieval models. In *European Conference on Information Retrieval*. Springer, 95–110.

[7] Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Boston, MA, USA) *(SIGIR '09)*. Association for Computing Machinery, New York, NY, USA, 758–759. doi:10.1145/1571941.1572114

[8] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Jimmy Lin. 2022. Overview of the TREC 2021 deep learning track. In *Text REtrieval Conference (TREC)*. NIST, TREC. https://www.microsoft.com/en-us/research/publication/overview-of-the-trec-2021-deep-learning-track/

[9] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, Jimmy Lin, Ellen M. Voorhees, and Ian Soboroff. 2023. Overview of the TREC 2022 deep learning track. In *Text REtrieval Conference (TREC)*. NIST, TREC. https://www.microsoft.com/en-us/research/publication/overview-of-the-trec-2022-deep-learning-track/

[10] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the TREC 2019 deep learning track. arXiv:2003.07820 [cs.IR]

[11] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, Ellen M. Voorhees, and Ian Soboroff. 2021. TREC Deep Learning Track: Reusable Test Collections in the Large Data Regime. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai (Eds.). ACM, 2369–2375. doi:10.1145/3404835.3463249

[12] Shuai Ding and Torsten Suel. 2011. Faster top-k document retrieval using block-max indexes. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011*, Wei-Ying Ma, Jian-Yun Nie, Ricardo Baeza-Yates, Tat-Seng Chua, and W. Bruce Croft (Eds.). ACM, 993–1002. doi:10.1145/2009916.2010048

[13] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (<conf-loc>, <city>Virtual Event</city>, <country>Canada</country>, </conf-loc>) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 2288–2292. doi:10.1145/3404835.3463098

[14] Artem Grotov and Maarten De Rijke. 2016. Online learning to rank for information retrieval: Sigir 2016 tutorial. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 1215–1218.

[15] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 2021. Efficiently teaching an effective dense retriever with balanced topic aware sampling. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 113–122.

[16] N. Jardine and Cornelis Joost van Rijsbergen. 1971. The use of hierarchic clustering in information retrieval. *Inf. Storage Retr.* 7, 5 (1971), 217–240. doi:10.1016/0020-0271(71)90051-9

[17] Shivaram Kalyanakrishnan, Ambuj Tewari, Peter Auer, and Peter Stone. 2012. PAC Subset Selection in Stochastic Multi-armed Bandits. *Proceedings of the 29th International Conference on Machine Learning, ICML 2012* 1 (01 2012).

[18] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, Online, 6769–6781. doi:10.18653/v1/2020.emnlp-main.550

[19] Hrishikesh Kulkarni, Nazli Goharian, Ophir Frieder, and Sean MacAvaney. 2024. LexBoost: Improving Lexical Document Retrieval with Nearest Neighbors. In *Proceedings of the ACM Symposium on Document Engineering 2024* (San Jose, CA, USA) *(DocEng '24)*. Association for Computing Machinery, New York, NY, USA, Article 16, 10 pages. doi:10.1145/3685650.3685658

[20] Hrishikesh Kulkarni, Sean MacAvaney, Nazli Goharian, and Ophir Frieder. 2023. Lexically-Accelerated Dense Retrieval. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023*, Hsin-Hsi Chen, Wei-Jou (Edward) Duh, Hen-Hsen Huang, Makoto P. Kato, Josiane Mothe, and Barbara Poblete (Eds.). ACM, 152–162. doi:10.1145/3539618.3591715

[21] Jurek Leonhardt, Koustav Rudra, Megha Khosla, Abhijit Anand, and Avishek Anand. 2022. Efficient Neural Ranking using Forward Indexes. In *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, Frédérique Laforest, Raphaël Troncy, Elena Simperl, Deepak Agarwal, Aristides Gionis, Ivan Herman, and Lionel Médini (Eds.). ACM, 266–276. doi:10.1145/3485447.3511955

[22] Shuai Li, Tor Lattimore, and Csaba Szepesvári. 2019. Online learning to rank with features. In *International Conference on Machine Learning*. PMLR, 3856–3865.

[23] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2021. In-Batch Negatives for Knowledge Distillation with Tightly-Coupled Teachers for Dense Retrieval. In *Proceedings of the 6th Workshop on Representation Learning for NLP, RepL4NLP@ACL-IJCNLP 2021, Online, August 6, 2021*, Anna Rogers, Iacer Calixto, Ivan Vulic, Naomi Saphra, Nora Kassner, Oana-Maria Camburu, Trapit Bansal, and Vered Shwartz (Eds.). Association for Computational Linguistics, 163–173. doi:10.18653/V1/2021.REPL4NLP-1.17

[24] Jing Lu, Ji Ma, and Keith B Hall. 2022. Zero-shot Hybrid Retrieval and Reranking Models for Biomedical Literature.. In *CLEF (Working Notes)*. 281–290.

[25] Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2024. Fine-tuning llama for multi-stage text retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2421–2425.

[26] Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. 2020. Expansion via Prediction of Importance with Contextualization. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, Jimmy X. Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu (Eds.). ACM, 1573–1576. doi:10.1145/3397271.3401262

[27] Sean MacAvaney and Nicola Tonellotto. 2024. A Reproducibility Study of PLAID. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2024)*. ACM, 1411–1419. doi:10.1145/3626772.3657856

[28] Sean MacAvaney, Nicola Tonellotto, and Craig Macdonald. 2022. Adaptive Re-Ranking with a Corpus Graph. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022*, Mohammad Al Hasan and Li Xiong (Eds.). ACM, 1491–1500. doi:10.1145/3511808.3557231

[29] Sean MacAvaney and Xi Wang. 2023. Online Distillation for Pseudo-Relevance Feedback. *CoRR* abs/2306.09657 (2023). doi:10.48550/ARXIV.2306.09657 arXiv:2306.09657

[30] Yury A. Malkov and Dmitry A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* 42, 4 (2020), 824–836. doi:10.1109/TPAMI.2018.2889473

[31] Irina Matveeva, Chris Burges, Timo Burkard, Andy Laucius, and Leon Wong. 2006. High accuracy retrieval with multiple nested ranker. In *SIGIR 2006: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA, August 6-11, 2006*, Efthimis N. Efthimiadis, Susan T. Dumais, David Hawking, and Kalervo Järvelin (Eds.). ACM, 437–444. doi:10.1145/1148170.1148246

[32] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016 (CEUR Workshop Proceedings, Vol. 1773)*, Tarek Richard Besold, Antoine Bordes, Artur S. d'Avila Garcez, and Greg Wayne (Eds.). CEUR-WS.org. https://ceur-ws.org/Vol-1773/CoCoNIPS_2016_paper9.pdf

[33] Rodrigo Frassetto Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document Ranking with a Pretrained Sequence-to-Sequence Model. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020 (Findings of ACL, Vol. EMNLP 2020)*, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, 708–718. doi:10.18653/V1/2020.FINDINGS-EMNLP.63

[34] Iadh Ounis, Gianni Amati, Vassilis Plachouras, Ben He, Craig Macdonald, and Douglas Johnson. 2005. Terrier information retrieval platform. In *Advances in Information Retrieval: 27th European Conference on IR Research, ECIR 2005, Santiago de Compostela, Spain, March 21-23, 2005. Proceedings 27*. Springer, 517–519.

[35] Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. Rankvicuna: Zero-shot listwise document reranking with open-source large language models.

arXiv preprint arXiv:2309.15088 (2023).

[36] Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. RankZephyr: Effective and Robust Zero-Shot Listwise Reranking is a Breeze! *arXiv preprint arXiv:2312.02724* (2023).

[37] Mandeep Rathee, Sean MacAvaney, and Avishek Anand. 2025. Quam: Adaptive Retrieval through Query Affinity Modelling. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining* (Hannover, Germany) *(WSDM '25)*. Association for Computing Machinery, New York, NY, USA, 954–962. doi:10.1145/3701551.3703584

[38] Clémence Réda, Emilie Kaufmann, and Andrée Delahaye-Duriez. 2021. Top-m identification for linear bandits. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 1108–1116.

[39] Revanth Gangi Reddy, Pradeep Dasigi, Md Arafat Sultan, Arman Cohan, Avirup Sil, Heng Ji, and Hannaneh Hajishirzi. 2023. Inference-time Re-ranker Relevance Feedback for Neural Information Retrieval. *arXiv preprint arXiv:2305.11744* (2023).

[40] Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.* 3, 4 (apr 2009), 333–389. doi:10.1561/1500000019

[41] Weiwei Sun, Zheng Chen, Xinyu Ma, Lingyong Yan, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Instruction Distillation Makes Large Language Models Efficient Zero-shot Rankers. *ArXiv* abs/2311.01555 (2023).

[42] Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is ChatGPT good at search? investigating large language models as re-ranking agents. *arXiv preprint arXiv:2304.09542* (2023).

[43] Lidan Wang, Jimmy Lin, and Donald Metzler. 2011. A cascade ranking model for efficient ranked retrieval. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. 105–114.

[44] Shuai Wang, Shengyao Zhuang, and Guido Zuccon. 2021. Bert-based dense retrievers require interpolation with bm25 for effective passage retrieval. In *Proceedings of the 2021 ACM SIGIR international conference on theory of information retrieval*. 317–324.

[45] Masrour Zoghi, Tomas Tunys, Mohammad Ghavamzadeh, Branislav Kveton, Csaba Szepesvari, and Zheng Wen. 2017. Online learning to rank in stochastic click models. In *International conference on machine learning*. PMLR, 4199–4208.