Quam

Adaptive Retrieval through Query Affinity Modelling

Rathee, Mandeep; MacAvaney, Sean; Anand, Avishek

DOI
10.1145/3701551.3703584

Publication date
2025

Document Version
Final published version

Published in
WSDM 2025

QUAM: Adaptive Retrieval through Query Affinity Modelling

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

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

A central task in information retrieval and the NLP communities is relevance modeling, which aims to rank documents based on their expressed information needs Many knowledge-intensive retrieval tasks are powered by a first-stage retrieval stage for context selection, followed by a more involved task-specific model. However, using this filtering (cascading) approach inherently limits the recall of subsequent stages. Recently, adaptive re-ranking techniques have been proposed to overcome this issue by continually selecting documents from the whole corpus, rather than only considering an initial pool of documents. However, so far these approaches have been limited to heuristic design choices, particularly in terms of the criteria for document selection. In this work, we propose a unifying view of the nascent area of adaptive retrieval by proposing QUAM, a *query-affinity model* of adaptive re-ranking that includes two complementary components: (1) a more principled algorithm for document selection, and (2) a data-driven approach to model document co-relevance during indexing. Our extensive experimental evidence shows that our proposed approach improves the recall performance by up to 26% over the standard re-ranking baselines. Further, the query affinity modelling and relevance-aware document graph components can be injected into any adaptive retrieval approach. The experimental results show the existing adaptive retrieval approach improves recall by up to 12%.

https://github.com/Mandeep-Rathee/quam

CCS Concepts

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

Keywords

neural re-ranking, adaptive retrieval, clustering hypothesis

ACM Reference Format:
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 (WSDM '25), March 10–14, 2025, Hannover, Germany.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3701551.3703584

WSDM '25, March 10–14, 2025, Hannover, Germany

ACM ISBN 979-8-4007-1329-3/25/03
https://doi.org/10.1145/3701551.3703584

1 Introduction

Relevance modelling, which estimates whether documents satisfy an information need provided by a query, is a central task in information retrieval and NLP. Many knowledge-intense tasks are powered by a first-stage retrieval/ranking stage for context selection, followed by a more involved task-specific model. Traditional ranking models that rely on lexical matching (e.g., BM25) are efficient and well-engineered based on decades of research, but they exhibit the well-known *vocabulary mismatch* problem due to the inherent under-specificity of queries. Recent methods based on dense retrieval [5] rely heavily on semantic similarity are slower and typically use a lossy approximate nearest neighbor search to achieve efficiency. In both ranking approaches, the common paradigm for ranking documents is based on the *retrieve and re-rank* paradigm; where a first stage retrieval (lexical or dense) is followed by a more involved re-ranking stage facilitated by a contextual transformer model. The primary objective of the first-stage retrieval is to maximize recall and efficiently filter out the most irrelevant documents. However, a major limitation of this paradigm is that the recall of the final result list is, by definition, bounded by the recall of the first-stage retrieval. In other words, documents filtered out by the first stage cannot appear in the re-ranked results.

To solve the bounded-recall problem, adaptive ranking techniques have been proposed that add additional opportunities to retrieve documents [6, 12]. The key idea of adaptive retrieval is based on modelling the similarities between documents in the corpus by constructing a *corpus graph* offline. During the re-ranking process, the neighbors of the top-scoring documents from the re-ranker are expanded using the corpus graph, allowing documents to be retrieved even if they were missed by the first-stage retriever. Adaptive re-ranking algorithms typically either alternate between scoring results from the first-stage and the corpus graph [12] or completely score the first-stage and then iteratively expand over the corpus graph [6]. Adaptive retrieval has shown to be successful with recall improvements of up to 11% for cross-encoders [12] and 15% for bi-encoders [6] when compared with existing methods and controlling for retrieval latency.

Limitations of current adaptive retrieval. However, there are two major limitations of existing adaptive retrieval approaches. Firstly, the quality of adaptive retrieval is based on the quality of the corpus graph, which has so far been constructed based on heuristic choices. Specifically, current corpus graphs encode lexical or semantic similarities between documents and are agnostic to query-based relevance. This results in corpus graphs considering documents that have high similarity to potentially non-relevant content and might not result in surfacing relevant documents. Secondly, adaptive re-ranking algorithms like GAR [12] do not consider

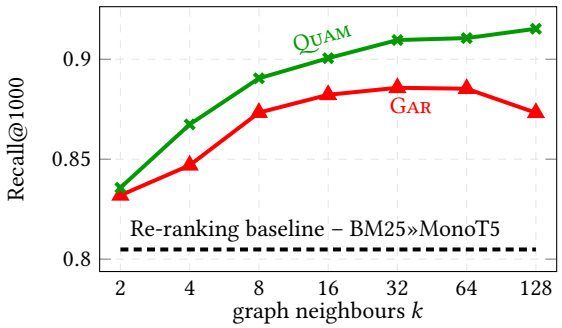

Figure 1: Recall comparison on the TREC DL20 dataset when the number of neighbors varies.

the degree of similarity between documents during the expansion process. Consequently, GAR cannot differentiate between the degrees of relatedness of the documents—only which documents are most similar. In an extreme case, consider a document that isn't related to any other document in the corpus; GAR will waste time by scoring the nearest neighbors of this document, even though none of them are even related to the original documents. This problem is further accentuated when denser corpus graphs or graphs with a larger number of neighbors. Figure 1 shows this phenomenon in action, with GAR's recall peaking at 32 neighbors per document, then dropping off as more are added.

Improved Corpus Graph Construction. In this work, we solve both the problems mentioned before by first improving the quality of the corpus graph. Toward this end, we train an edge-prediction model that predicts whether there should be an edge between a pair of potentially relevant documents by exploiting co-relevance information in ranking datasets. This learnt model as an additional output also provides an edge weight based on the *learnt affinity* between documents that we refer to as the learnt-affinity scores or LAFF scores in short. Using learnt affinities, we are able to prune the original corpus graphs, leading to potential efficiency gains. The affinity scoring can also be leveraged for better candidate selection to improve overall recall.

Query processing using Affinity modelling. Our second contribution is to propose an adaptive retrieval strategy called QUAM (short for query affinity modelling) that judiciously chooses the neighborhood documents to re-rank by exploiting the affinity scores or edge weights. Unlike GAR which does not differentiate between neighbors of a relevant document, we propose a *query-affinity model* that exploits the relevance-aware document affinity graph.

Experimental Evaluation. We conduct an extensive experimental evaluation on TREC-DL '19 and '20 passage re-ranking tasks under multiple scenarios to show the efficacy and effectiveness of QUAM. Our results show that we can outperform the baselines resulting in clear recall improvements by up to 26% and GAR by up to 12%. Secondly, we show that our corpus graphs encode affinity scores that help not only QUAM but also existing algorithms like GAR. GAR improves by up to 9% when using our corpus affinity graphs. Finally, we show that QUAM is robust to dense corpus graphs (see Figure 1) in that it can effectively choose between relevant and non-relevant neighbors, unlike GAR, which adds additional noise with increasing graph neighborhoods.

Contributions.

- We propose a novel approach to construct a corpus graph that faithfully encodes the co-relevance relations between documents called as the document affinity graph.
- We provide concrete instantiation and a principled algorithm QUAM for adaptive query processing.
- We conduct extensive experimentation to show that we can outperform existing static and adaptive retrieval baselines.

2 Background and Related Work

Based on the long-standing Probability Ranking Principle [24], most contemporary search systems use a relevance model $\phi(q, d)$ that provides a real-valued estimate of the relevance of a document (d) to a query (q). The goal of a retrieval engine is to identify the top k documents with the highest relevance score R from a large corpus of documents C.

The trivial solution to this problem is an exhaustive search, which scores and sorts all documents in C. This approach is inherently unscalable since the cost increases linearly with the size of the corpus. Some types of relevance models, particularly those that use lexical [25] (and recently learned sparse [17]) representations to calculate relevance, can leverage their sparsity with inverted index data structures [30] and algorithms (e.g., MaxScore [26] and Block-MaxWAND [3]) to reduce the cost of retrieval. These approaches are able to guarantee that the *exact* top k results from the corpus are returned since the relevance models behave predictably over their representations (e.g., they can often be reduced to a simple dot product between the sparse representation of the query and document).

For other types of relevance models, there are no known approaches to guarantee an exact set of top k results faster than an exhaustive search. For these types of models, an *approximation* of the top results is often used instead. For instance, engines that use dense bi-encoder models [5] often use algorithms such as HNSW [14] to perform an approximate search. HNSW builds a neighborhood graph, where each document is linked to an approximation of its nearest neighbors. By using a hierarchy of these neighborhoods with progressively smaller subsets of the corpus, HNSW is able to scan the hierarchy to find an approximation of the most similar documents to a query vector. This technique relies on the long-standing Cluster Hypothesis [4], which suggests that relevant documents are likely to be near other relevant ones.

Still, approaches like HNSW are not universally applicable since they rely on scoring vectors using a simple, well-behaved function (e.g., a dot product). Many relevance models do not have this quality. For instance, learning-to-rank models often use a tree-based decision function over features [28], and neural cross-encoder models estimate relevance through a complicated combination of query and document signals [19]. The typical approach in this setting is to perform re-ranking [16] (also called cascading in literature), wherein an initial pool of $k' \geq k$ documents is retrieved using a "first-stage" scalable approach (e.g., using the sparse or dense vector methods outlined above), and another relevance model (e.g., a learning-to-rank model or cross-encoder) then re-orders only those

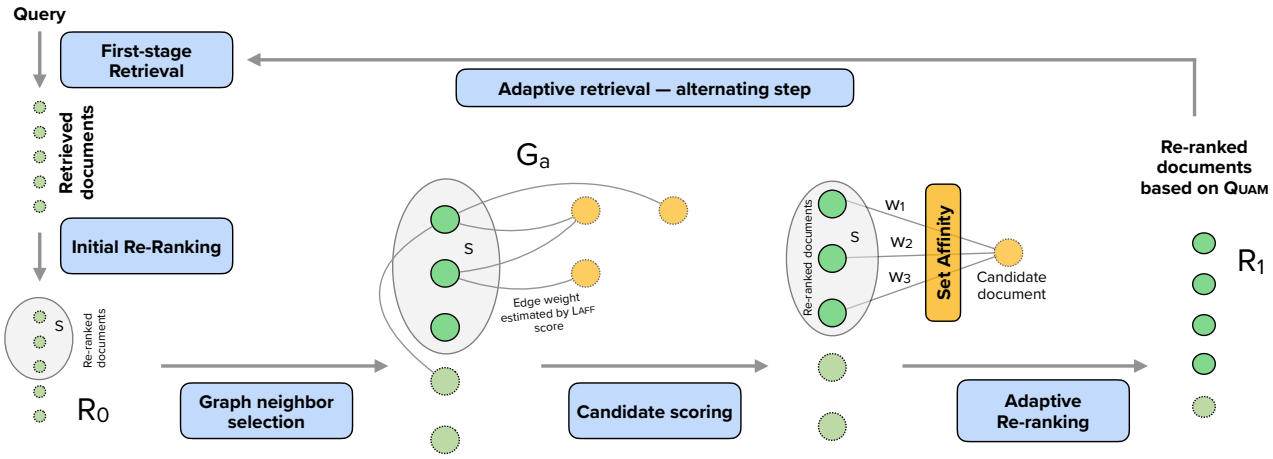

Figure 2: An overview of the adaptive retrieval through the query affinity modelling QUAM. The W_is represent the affinity or edge weights.

documents. Remarkably, re-ranking is not only helpful for learning-to-rank and cross-encoder models, but also highly competitive with HNSW for dense bi-encoder models, given the high efficiency of lexical retrieval [7, 8, 27]. The key limitation of re-ranking is that it bounds performance by the documents recalled in the first stage; if a relevant document is not retrieved at that stage, it cannot be re-ranked in the final results. This limitation can be particularly problematic when the first stage uses only lexical signals, since one of the main benefits of models like cross-encoders is to overcome lexical mismatches.

Adaptive re-ranking techniques have been proposed to overcome the recall limitation of re-ranking [12]. By using a corpus graph and by leveraging document relevance estimations obtained during the re-ranking process, adaptive re-ranking retrieves and scores documents that were not returned during first-stage retrieval. Similar to HNSW, adaptive re-ranking leverages the Cluster Hypothesis by suggesting that documents nearby ones with high relevance estimations may also score highly. It overcomes the recall problem of re-ranking by not relying exclusively on the first stage results but instead also leveraging pseudo-relevance signals obtained during the re-ranking process itself. Several adaptive re-ranking strategies have been proposed. For instance, GAR [12] alternates scoring batches between the initial pool of documents and those obtained from the corpus graph. Follow-up work from the GAR authors suggests that this *alternating* strategy is comparable with other simple strategies for document selection from a corpus graph [11]. Beyond cross-encoders, adaptive re-ranking approaches have also been effectively applied to bi-encoders [6, 10] and ensemble models [29]. For instance, LADR [6] leverages an efficient lexical model to identify good "seeds" to further explore its corpus graph.

We contrast our work with prior work in two main ways. First, we replace the heuristic-based approaches of document selection, such as the alternating approach in GAR, with query affinity modelling to be more principled when selecting documents for scoring. Second, we replace the neighborhood graph construction process with a new learned document-document affinity model. Together, we find these changes provide more favorable selection criteria for documents and ultimately yield improved efficiency-effectiveness trade-offs.

3 Query Affinity Modelling

In the following subsections, we focus on our proposed approach called QUAM, specifically the two main components: (1) the document affinity graph G_a based on LAFF scores, and (2) the query affinity modelling based on SETAFF scores. Finally, we provide a principled algorithm for using QUAM in an adaptive retrieval pipeline. An overview of our approach is presented in Figure 2.

3.1 Document Affinity Graph

Our main objective in this section is to construct document affinity graphs (in reasonable time) that help estimate relevance accurately while also improving query processing efficiency. Using the notation presented in Section 2, we consider *relevance* $\phi(q, d)$ a relevance model ϕ's estimation of the relevance of document d to the information need expressed by query q. In contrast, we define *affinity* as the degree of association between *documents* that models co-relevance. In other words, a pair of documents should have high affinity if two documents can satisfy similar information needs and low affinity if they cannot. Existing adaptive retrieval approaches use an unweighted top-k similarity graph (called a corpus graph) denoted by G_c. In the case of dense retrieval, a corpus graph, can be constructed from the representation space induced by the trained document encoders. For sparse retrieval, the corpus graph can be constructed using a document as a query. The top-k-ranked result documents are its k-nearest neighbors.

To construct this *affinity* graph G_a, we start from an initial corpus graph, G_c, of the corpus documents and learn a model that predicts the *affinity* for each of the edges in this corpus graph. For this, we source the training data that uses *query-document-label* triples to construct co-relevant document pairs that share the same query. Consequently, we train a model f (learnt-affinity model) that learns *affinity* score between the pairs of documents. The affinity or edge

Table 1: Number of queries by the number of relevant documents labeled in MSMARCO passage train dataset.

#rel docs	1	2	3	4	5	6	7
#query	477580	21868	2718	612	131	22	8

weight between a pair of documents d_i and d_j in G_a is denoted by $f(d_i, d_j)$. Theoretically, we could use any text-matching model for f, though in this work, we use a cross-encoder between the two documents, given their high effectiveness at relevance modelling.

3.2 Training Data for Learnt Affinity Model

In some cases, large-scale datasets may already contain human-annotated co-relevance labels based on documents that are labeled as relevant to the same query. However, in practice, we posit that co-relevance labels will be sufficiently rare for training an affinity model. Indeed, Table 1 shows that less than 5% of queries in the popular MSMARCO passage train dataset [18] have positive relevance labels to more than one passage, given only around 25k queries available for training. This lack of true co-relevant pairs motivates us to source pseudo co-relevant pairs.

For each query, we source k positive and negative documents. We start with a standard re-ranking approach, such as a retriever, followed by a ranker. Initially, the retriever retrieves the initial pool of documents, and then the ranker is applied to re-rank this pool. Let R_0 denote the pool of documents retrieved by the retriever, and R_1 denote the pool after it has been re-ranked by the ranker. Let $\mathcal{P}_q$ and $\mathcal{N}_q$ denote the set of relevant and non-relevant documents for the query q. We choose the top k documents from R_0 as set $\mathcal{P}_q$ and the last k documents as set $\mathcal{N}_q$ and the top k documents from R_1 as set S. Finally, for each $p \in \mathcal{P}_q$, $n \in \mathcal{N}_q$, and $d \in S$ we have $(p, d, 1) \in \mathcal{D}$ and $(n, d, 0) \in \mathcal{D}$.

We fine-tune a bert-base[1] model (as a cross-encoder) on the training Data $\mathcal{D}$ by minimizing the binary cross-entropy loss.

$$L(\mathcal{D}) = -\frac{1}{|\mathcal{D}|} \sum_{(x,d,y)\in\mathcal{D}} [y \log(f(x,d)) + (1-y)\log(1-f(x,d))] \tag{1}$$

Further details on training are in Section A of supplementary material [2] available with our code. Finally, we use the model f to create the affinity graph G_a by re-scoring each value in an existing corpus graph G_c.

3.3 Query Processing Using Quam

Given a document affinity graph G_a, the query affinity model intends to characterize the affinity of any document to a ranked set of documents S. We define the expected set affinity (or SetAff in short) of a document d from an affinity graph G_a to a set of ranked documents S as

$$\textsc{SetAff}(d, S) = \sum_{d' \in S} P(Rel(d')) \cdot f(d, d') \tag{2}$$

[1] https://huggingface.co/google-bert/bert-base-uncased
[2] https://github.com/Mandeep-Rathee/quam/blob/main/appendix.pdf

Algorithm 1 Adaptive Retrieval Using Quam

Input: Initial ranking R_0, batch size b, budget c, affinity graph G_a, top re-ranked documents s
Output: Re-Ranked pool R_1

```
R_1 ← ∅                                          ▷ Re-Ranking results
P ← R_0                                          ▷ Re-ranking pool
F ← ∅                                            ▷ Graph frontier
S ← ∅                                            ▷ top ranked documents
do
    B ← Score(top b from P, subject to c)        ▷ Using monoT5
    R_1 ← R_1 ∪ B                                ▷ Add batch to results
    R_0 ← R_0 \ B                                ▷ Discard batch from initial ranking
    F ← F \ B                                    ▷ Discard batch from frontier
    S ← Select(top s from R_1)                   ▷ Select top s ranked docs
    F ← F ∪ (Neighbours(B ∩ S, G_a) \ R_1)       ▷ Update frontier
    F ← SetAff(d, S)   ∀d ∈ F                    ▷ Assign set affinity scores
    P ← { R_0  if P = F
        { F    if P = R_0                        ▷ Alternate initial ranking and frontier
while |R_1| < c
```

where $P(Rel(d'))$ encodes the estimated relevance distribution induced by a relevance scorer (e.g., MonoT5 [20]) model ϕ. $P(R(d'))$ can be estimated in multiple ways. We let $P(Rel(d')) = \frac{e^{\phi(q,d)}}{\sum_{d'\in S} e^{\phi(q,d')}}$.

There are multiple methods that can be envisioned to estimate the $P(Rel(d'))$. For example, one could use retrieval scores, normalized ranked positions, or re-ranking scores. We posit that since d' is already re-ranked (i.e., $d' \in S$), we can use the re-ranking scores as better relevance estimates in comparison to retrieval scores or rank positions. We test this hypothesis empirically and find that using re-ranking scores for calculating the SetAff yields superior performance in comparison to using retrieval scores. We report the results for this ablation of the effect of retriever and ranker scores in Section B of the supplementary material available with our code.

For a given query q, let R_0 be the pool of initial ranked documents and R_1 be the re-ranked pool. As shown in Figure 2, the top documents from R_0 are used to explore the neighborhood documents in the affinity graph. These neighborhood candidate documents are assigned with set affinity scores using Equation 2. The high-set affinity candidate documents are selected for ranking and added to the re-ranked pool R_1. We keep alternating between the initial ranking and the neighbors to select the documents for re-ranking until we reach the re-ranking budget.

3.4 Adaptive Retrieval Using Quam

Algorithm 1 illustrates how we perform adaptive retrieval using our Quam model. Quam takes as input the initial rank pool R_0, a batch size b, a budget c, and an affinity graph G_a. Let F, initially empty, be the frontier that stores potential candidate documents for selection in R_1. Let P be the re-ranking pool, initialized with R_0. Let S be the set of top s (a hyper-parameter) re-ranked documents from R_1. We start with selecting top b (batch size) documents from R_0 and get relevance scores by using the Score( ) function. These b documents are added to the re-ranked pool R_1 and removed from

R_0. Next, we select the set S as top-s re-ranked documents from R_1, i.e., the top s documents we have re-ranked so far. Now, we use the *affinity graph*, G_a, to explore the neighborhood documents (excluding the ones already in R_1) of the documents that are newly added to the set S. We limit the neighbor lookup to only s documents because as the size of R_1 increases, calculating the SetAff scores becomes computationally expensive. The neighborhood documents are added to the frontier F. For each document d in the frontier F, the set affinity score to the set S (SetAff(d, S)) is calculated using Equation 2. In contrast to Gar, which considers all neighbors of the source document equally important, we use these set affinity scores to prioritize the documents in the frontier F.

Next, we choose the top b documents from this frontier F. In subsequent rounds, we alternate between R_0 and the frontier F similar to Gar [12] until the budget criteria are fulfilled.

4 Experimental Setup

We conduct a series of experiments to answer the following research questions:

- **RQ1** What is the impact of Quam on retrieval effectiveness compared to typical re-ranking and Gar?
- **RQ2** How helpful is the affinity-based graph G_a or Laff scores in prioritizing the neighbors for adaptive retrieval?
- **RQ3** What is the effect of graph depth k on adaptive retrieval methods?
- **RQ4** How efficient is Quam in comparison to the Gar and standard re-ranking pipelines?

4.1 Datasets and Evaluation

We conduct our experiments mainly on the TREC Deep Learning 2019 (DL19) and 2020 (DL20) datasets [2] which share MSMARCO passage corpus of 8.8M passages [18]. We validate our method on the DL19 and test on the DL20. The DL19 (validation) dataset consists of 43 queries with an average of 215 assessments per query. The DL20 (test) dataset consists of 54 queries with 211 relevance assessments per query. To evaluate the effectiveness of our approach, we use the nDCG@10, nDCG@c, and Recall@c, where c is the budget. We utilize the corpus graphs generated from a sparse retriever, BM25 [25], and a dense retriever, TCT [9], reusing the corpus graphs created by Gar.

4.2 Ranking Models and Baselines

For our experiments, we use different retrieval and ranking models and the most comparable adaptive retrieval baseline.

4.2.1 Retrieval Methods. We use both sparse (BM25) and dense (TCT) retrieval models. **BM25** is a sparse retrieval method based on the query terms present in the documents. We use top $c \in [50, 100, 1000]$ results from BM25 using a PISA [15] index. We use default parameters for retrieval. **TCT** is a dense retrieval model, a distilled version of the ColBERT model. We retrieve (exhaustively) top $c \in [50, 100, 1000]$ documents using TCT-ColBERT-HNP[3] [9].

4.2.2 Ranking Model. We use MonoT5 [20][4] as a ranker. MonoT5 is a variant of the T5 [23] model, which takes a query and document as input and generates a relevance score. This score is used to re-rank the documents. In our experiments, we use the MonoT5-Base (with 223M parameters) variant trained on the MS MARCO dataset. We denote it as MonoT5 for convenience.

[3]https://huggingface.co/castorini/tct_colbert-v2-hnp-msmarco
[4]https://huggingface.co/castorini/monot5-base-msmarco

Table 2: Hyper-parameters and their description.

Notation	Description
b	batch size
c	re-ranking budget
k	depth of the graph (number of neighbors to explore)
s	number of the top re-ranked documents from R_1

4.2.3 Baseline. We use the Graph Adaptive Retriever or Gar [12] as a baseline to compare Quam. Gar is an adaptive re-ranking approach that alternates between initial retrieved documents and neighbors of these documents in the corpus graph. Given the source document and its relevance score, Gar assigns the same score to all its neighbors to prioritize them.

We conduct all experiments using PyTerrier [13] framework. We follow the pipeline's notations from PyTerrier. For example, the pipeline "BM25»MonoT5" retrieves using BM25 and re-ranks them using the MonoT5 model.

4.3 Other Hyper-parameters

Table 2 shows the hyper-parameters and their corresponding description. For Table 3, we choose batch size b=16, graph depth k=16, and vary budget c from 50, 100, and 1000 and select S with s=10, 30, and 300 respectively. We explore the robustness of our proposed method by varying batch size b and graph depth k in [2,128] (by power of 2).

5 Results and Analysis

In this section, we discuss the results and analysis of our experiments. In all our experiments, we denote the vanilla graph-based adaptive retrieval [12] by Gar and the corresponding type of corpus graph indicated in subscript, for instance, Gar_{BM25} represent the graph-based adaptive retrieval method Gar when BM25-based corpus graph (G_c) is used. We denote our query affinity model as Quam, with the type of affinity graph (G_a) indicated in subscript. For instance, $\text{Quam}_{\text{BM25}}$ represents the query affinity model with the BM25-based affinity graph (i.e., the Laff scores from the model f between pairs of documents are used to calculate the SetAff scores (Equation 2)).

5.1 Effectiveness of Quam

To answer **RQ1**, we assess the effectiveness of Quam by analyzing its impact on re-ranking pipelines with sparse (BM25) and dense (TCT) retrievals followed by a scoring function (MonoT5). We report the performance of Quam in Table 3 on the TREC DL 19 and 20. We incorporate lexical (BM25) and semantic (TCT) based corpus graphs. We compare our approach Quam with standard re-ranking pipelines and Gar. We vary re-ranking budgets c to 50, 100, and 1000. Each row in Table 3 represents a ranking system.

The standard ranking pipelines (retriever»MonoT5) are shown in gray color. Additionally, we include the MonoT5 exhaustive (in short MonoT5-Exh.) search results for both TREC DL19 and 20. The

Table 3: Effectiveness of GAR and QUAM on TREC DL19 and 20 dataset. Significant improvements (paired t-test, $p < 0.05$, using Bonferroni correction) with the re-ranking baseline (retriever»MonoT5) and GAR are marked with B and G respectively in the superscript. The best result for each pipeline is 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	MonoT5-Exh.	0.672	0.625	0.512	0.672	0.611	0.599	0.672	0.691	0.834
	BM25»MonoT5	0.676	0.546	0.389	0.696	0.571	0.497	0.724	0.696	0.755
	w/ GAR_{BM25}	0.694	0.573	0.426	0.716	0.605	0.547	0.719	0.736	^{B}0.833
	w/ $\text{QUAM}_{\text{BM25}}$	**0.706**	^{BG}**0.615**	^{BG}**0.480**	**0.720**	^{BG}**0.651**	^{BG}**0.611**	**0.732**	^{B}**0.758**	^{BG}**0.867**
	w/ GAR_{TCT}	**0.724**	^{B}**0.620**	^{B}0.476	^{B}**0.747**	^{B}**0.656**	^{B}**0.606**	**0.734**	^{B}**0.754**	^{B}**0.859**
	w/ QUAM_{TCT}	0.704	^{B}0.612	^{B}**0.481**	0.722	^{B}0.642	^{B}0.601	0.721	^{B}0.747	^{B}0.857
	TCT»MonoT5	0.732	0.647	0.506	**0.722**	0.638	0.610	**0.699**	0.704	0.830
	w/ GAR_{BM25}	0.738	0.639	0.515	0.721	0.642	0.626	**0.699**	^{B}0.740	^{B}0.891
	w/ $\text{QUAM}_{\text{BM25}}$	**0.743**	^{BG}**0.684**	^{B}**0.556**	**0.722**	^{BG}**0.678**	^{B}**0.670**	0.696	^{B}**0.741**	^{B}**0.896**
	w/ GAR_{TCT}	0.732	0.658	0.534	**0.721**	0.653	0.638	**0.697**	0.722	0.860
	w/ QUAM_{TCT}	**0.740**	^{B}**0.673**	**0.538**	**0.721**	^{B}**0.667**	**0.659**	0.692	^{B}**0.728**	**0.881**
DL20	MonoT5-Exh.	0.649	0.592	0.576	0.649	0.593	0.670	0.649	0.682	0.852
	BM25»MonoT5	0.660	0.549	0.465	0.675	0.574	0.569	**0.716**	0.710	0.805
	w/ GAR_{BM25}	0.679	0.569	0.501	0.703	^{B}0.603	0.607	0.711	^{B}0.748	^{B}0.882
	w/ $\text{QUAM}_{\text{BM25}}$	^{BG}**0.716**	^{BG}**0.617**	^{BG}**0.558**	^{B}**0.715**	^{BG}**0.645**	^{BG}**0.664**	0.707	^{B}**0.755**	^{B}**0.901**
	w/ GAR_{TCT}	^{B}0.720	^{B}0.617	^{B}0.570	**0.718**	^{B}0.650	^{B}0.688	0.699	0.740	^{B}0.894
	w/ QUAM_{TCT}	^{B}**0.727**	^{B}**0.628**	^{B}**0.575**	0.713	^{B}**0.652**	^{B}**0.696**	**0.703**	**0.751**	^{B}**0.900**
	TCT»MonoT5	0.722	0.642	0.652	0.701	0.627	0.713	**0.672**	0.691	0.848
	w/ GAR_{BM25}	**0.724**	0.640	0.637	**0.717**	0.643	0.730	0.669	^{B}0.720	^{B}0.891
	w/ $\text{QUAM}_{\text{BM25}}$	0.720	^{BG}**0.664**	**0.660**	0.709	^{BG}**0.669**	^{B}**0.755**	0.670	^{BG}**0.732**	^{B}**0.916**
	w/ GAR_{TCT}	**0.722**	0.658	0.647	0.702	0.643	0.729	**0.669**	^{B}0.707	0.868
	w/ QUAM_{TCT}	0.720	^{B}**0.664**	**0.658**	**0.708**	^{BG}**0.663**	^{B}**0.750**	**0.669**	^{BG}**0.720**	^{BG}**0.887**

recall difference between MonoT5-Exh. and the standard re-ranking pipeline BM25»MonoT5 indicates that the BM25 retrieval fails to retrieve relevant documents that MonoT5 is capable of ranking well and hence does a poor job approximating a full MonoT5 search. This observation highlights the potential for further improvements in retrieval performance using, for instance, adaptive re-ranking techniques.

We observe the significant improvements by QUAM with the affinity graph from both BM25 and TCT over the standard ranking baselines across different budget sizes. The most substantial recall improvements can be seen with a low re-ranking budget, and hence, the improved recall results in better ranking performance. In particular, in comparison to the BM25»MonoT5 pipeline, $\text{QUAM}_{\text{BM25}}$ improves the recall@50 from 0.389 to 0.480 (23.39%) on DL19 and from 0.465 to 0.558 (20%) on DL20. Similarly, $\text{QUAM}_{\text{BM25}}$ improves the nDCG@50 by 12.64% on DL19 and 12.39% on DL20. $\text{QUAM}_{\text{BM25}}$ shows similar trends as we increase the budget size. In addition, the QUAM_{TCT} improves the recall@50 from 0.465 to 0.588 (26.45%) and recall@100 from 0.569 to 0.696 (22.32%) on DL20. It is important to note that the improvements made by GAR_{BM25} over the standard re-ranking baseline are not statistically significant, particularly at budget c=50 and 100. However, QUAM demonstrates significant improvements over GAR when a sparse (BM25) retriever in combination with the BM25-based graph. In particular, $\text{QUAM}_{\text{BM25}}$ improves recall@50 from 0.426 to 0.480 (12.68%) on DL 19 and from 0.569 to 0.617 (11.38%) on DL 20. For a dense (TCT) retriever, the $\text{QUAM}_{\text{BM25}}$ significantly improves nDCG@50 and nDCG@100.

Surprisingly, the $\text{QUAM}_{\text{BM25}}$outperforms the expensive MonoT5 exhaustive pipeline. In particular, $\text{QUAM}_{\text{BM25}}$ improves recall@100 from 0.599 to 0.611 (2%), and nDCG@100 from 0.611 to 0.651 (6.55%), and nDCG@10 from 0.672 to 0.720 (7.14%) on DL19. The TCT-based affinity graph also leads to similar trends. We note that MonoT5 is not an oracle relevance model; it can mistakenly assign high relevance scores to non-relevant documents. An exhaustive search setting maximizes the chances of retrieving these non-relevant documents since all documents are scored, ultimately reducing effectiveness. Meanwhile, adaptive re-ranking systems inherently constrain the search space through the initial pool and corpus graph, thereby reducing the chance of encountering this noise and resulting in higher effectiveness.

QUAM_{TCT} does not show significant improvements over GAR_{TCT} (except TCT»MonoT5 on DL20), but the performance remains comparable. Also, the GAR_{TCT} does not show significant improvements over TCT»MonoT5 pipeline, however, the QUAM_{TCT} can achieve significant improvements, especially in terms of nDCG@c. It is also important to compare both adaptive retrieval approaches with exhaustive search results. The lack of significant improvements

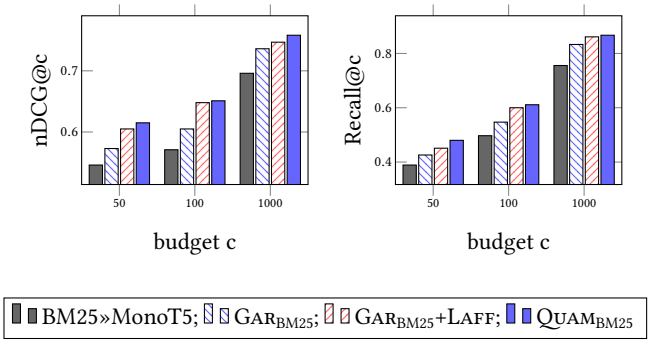

Figure 3: Effect of Learnt Affinity (LAFF) scores on adaptive retrieval on the DL19 dataset.

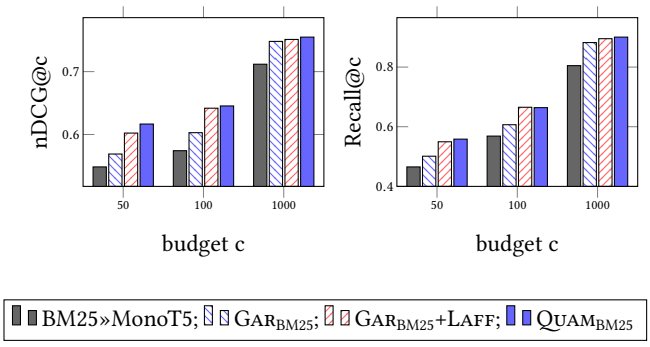

Figure 4: Effect of Learnt Affinity (LAFF) scores on adaptive retrieval on the DL20 dataset.

of QUAM$_{TCT}$ over GAR$_{TCT}$ could be due to the upper bound of the MonoT5 scoring function.

5.2 Effect of the Affinity Graph

To answer **RQ2**, we assess the effect of the LAFF scores on adaptive retrieval methods. Specifically, we want to see the impact of affinity graphs G_a provided to QUAM and GAR. Towards this, we inject the BM25-based affinity graph to GAR which we denote by GAR$_{BM25}$+LAFF. An affinity graph with GAR adds no computational overhead since the affinity scores are pre-computed.

We show the effectiveness of LAFF in Figure 3 (on DL19) and 4 (on DL20). The GAR$_{BM25}$+LAFF shows improvements over vanilla GAR$_{BM25}$, especially at lower re-ranking budgets. For instance, the GAR$_{BM25}$+LAFF improves recall@50 from 0.426 to 0.451 (5.87%) on DL19 and 0.501 to 0.549 (9.58%) on DL20. Similarly, it improves the nDCG@50 by 5.58% on DL19 and 7.1% on DL20. We observe similar trends at budget c=100.

5.3 Effect of Graph Depth

Like in standard retrieval settings, where recall improvements can be achieved by processing higher retrieval depths, in adaptive retrieval, higher recall can be achieved by processing deeper graph neighborhoods. However, traversing more documents either by accessing higher retrieval depths or graph neighborhoods adds more non-relevant documents that need to be differentiated from the relevant documents. In this experiment, we closely examine the ability of adaptive retrieval methods to achieve higher recall by traversing deeper graph neighborhoods. Towards this, in addition to GAR, we re-inforce GAR with components of our QUAM model – the LAFF scores and dynamic set affinity computation–to construct even stronger baselines. In Figure 5, we show the effect of graph depth k (in the first row) to show the performance of GAR, QUAM and the two baselines GAR+SETAFF and GAR+LAFF.

We firstly observe that there is a noticeable difference in performance between GAR and QUAM at all graph depths and is magnified at higher graph depths. In fact, the performance of GAR degrades substantially at higher graph depths due to its inability to differentiate between relevant or non-relevant documents. It is also clear that LAFF scores have a positive impact on GAR (similar to our observation in the last section). However, even GAR+LAFF degrades in performance at higher graph depths. This is mainly due to the fact that GAR cannot differentiate between two neighbors of the same relevant document. Secondly, GAR also processes all neighbors of a re-ranked document before going to the next relevant document, introducing the risk of adding potentially non-relevant documents if the ranked document is also non-relevant. Finally, we can also see that when GAR is re-inforced with the careful SETAFF selection in the GAR+SETAFF baseline, it is able to source from more relevant neighbors. However, the inability of GAR to differentiate between two neighbors of the same relevant document means it still underperforms QUAM at higher retrieval depths.

We also show the effect of the graph depth on ranking performances across different budgets in Figure 7 (on DL19) and 8 (on DL20) in Section C of supplementary material.

Interestingly, all approaches show insensitivity to variations in batch size. This characteristic of batch size insensitivity is advantageous as it enables the utilization of the full computational capacity of the hardware, consequently reducing latency. We show the effect of batch size on the ranking performances across different budgets in Figure 9 (on DL19) and 10 (on DL20) in Section D of supplementary material.

5.4 Efficiency of Query Processing

We re-iterate that re-ranking pipelines using adaptive retrieval have the same number of re-ranking operations as classical re-ranking pipelines, and this cost dominates the total computational cost of the pipeline. Indeed, adaptive retrieval procedures like GAR and QUAM are designed to contribute minimally to the total computational cost. To verify this property empirically, we performed latency experiments to assess the computational overhead introduced by QUAM in comparison to GAR. Note that while GAR indiscriminately schedules candidate documents for re-ranking, QUAM selects documents by computing a set-affinity score for each candidate document.

For a fair comparison between GAR and QUAM, we use the same MonoT5 re-ranker, a BM25-based graph of depth $k = 16$ and a batch $b = 16$ on the same hardware. While the MonoT5 scoring process leverages a GPU for hardware acceleration, both GAR and QUAM utilize only a single thread on CPU. In Table 4, we report the recall and mean latency (in ms) per query at different re-ranking budgets. For stable measurements, we take the average over 5 consecutive

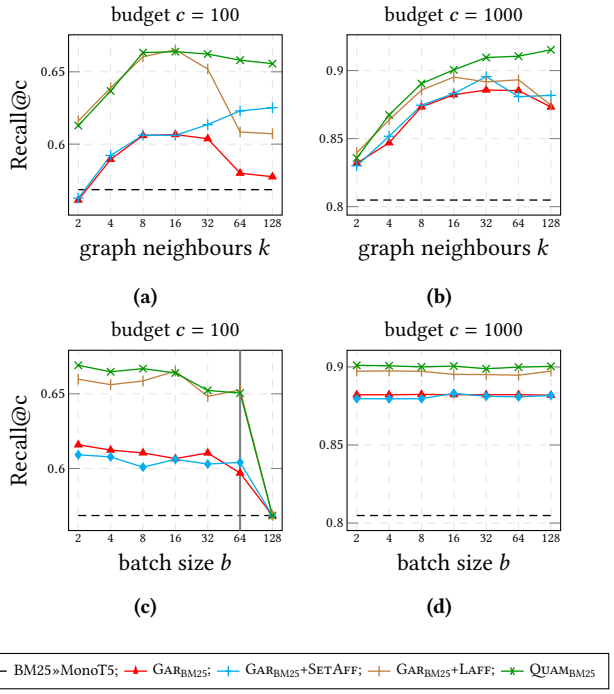

Figure 5: Recall comparison on the TREC DL20 dataset when the number of neighbours k (with fixed $b = 16$) and batch size b (with fixed $k = 16$) vary. The vertical line at $b = 64$ separates the region where $b > c$.

runs. We find that the variance in runtimes is as low as 0.01 ms, and hence we discard it in the table.

We first observe that both adaptive retrieval-based approaches, GAR and QUAM account for only 2 – 3% of the total time taken by ranker (MonoT5). For instance, the MonoT5 takes an average of approximately 3479.66 ms (with a batch size of 64) to re-rank 1000 documents per query on our hardware. On the other hand, the adaptive retrieval components of GAR and QUAM (with batch size b=16) take around 97.05 and 95.72 ms, respectively.

At lower re-ranking budgets ($c = 50$ and $c = 100$), the QUAM takes slightly longer (an additional time of 0.74 and 1.97 ms, but with recall improvement of 12.7% and 11.7% respectively) to process the neighborhood documents. Since the GAR looks for neighbors of b documents at each iteration, on the other hand, QUAM looks for neighbors of $|S| = s$ documents and uses the Equation 2 to compute the SETAFF scores. However, as the budget c increases, QUAM also outperforms GAR in terms of speed since the number of lookups for QUAM is less than that of GAR. The most important observation is that the QUAM achieves the recall of 0.849 in 57.36 ms/query whereas GAR takes 97.05 ms/query to obtain a recall of 0.833. This demonstrates that, for a given sufficient budget size, QUAM outperforms GAR in both the quality and latency of adaptive retrieval. In conclusion, we see that the computational overheads for QUAM are comparable or sometimes better than GAR while delivering consistently better recall at all re-ranking budgets.

Table 4: Mean latency overheads for QUAM and GAR (ms/query). $|S|$ denotes the size of set S in Equation 2. We denote the gain/drop in performance by QUAM by a green/red triangle over the GAR.

		time (ms)		Recall@c	
c	$\|S\|$	GAR$_{BM25}$	QUAM$_{BM25}$	GAR$_{BM25}$	QUAM$_{BM25}$
50	10	2.64	3.38(▼28%)	0.426	0.480(▲12.7%)
100	30	5.54	7.51(▼35.6%)	0.547	0.611(▲11.7%)
250	50	19.55	16.78(▲14.2%)	0.693	0.742(▲7.1%)
500	100	44.45	36.06(▲18.9%)	0.772	0.821(▲6.3%)
750	150	69.77	57.36(▲17.8%)	0.811	0.849(▲4.7%)
1000	300	97.05	95.72(▲1.4%)	0.833	0.867(▲4.1%)

6 Conclusion and Outlook

In this paper, we advance the new area of adaptive retrieval as a recall-improving approach for ad-hoc retrieval. We improve on the heuristic graph construction approaches used in earlier works by constructing a data-driven affinity graph with learned edge weights based on co-relevance information. Additionally, we propose a more principled adaptive retrieval algorithm (QUAM) that effectively chooses potential relevant documents from the affinity graph. Our experiments clearly show that our affinity modeling for graph construction and query processing improves not only our proposed approaches but also existing adaptive retrieval approaches. Secondly, we show that QUAM is able to judiciously filter out non-relevant documents, resulting in higher recall at deeper graph neighborhoods. Finally, we show that QUAM has a low computational overhead in comparison to GAR and can be used in many low-latency use cases. In the future, it would be important to extend adaptive retrieval techniques to scenarios where LLMs are either used to re-rank [21, 22], or in interactive query understanding [1]. Additionally, we would want to explore if one can further optimize the choice of candidate documents at low retrieval depths.

References

[1] Avishek Anand, Abhijit Anand, Vinay Setty, et al. 2023. Query understanding in the age of large language models. *arXiv preprint arXiv:2306.16004* (2023).
[2] 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. https://doi.org/10.1145/3404835.3463249
[3] 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. https://doi.org/10.1145/2009916.2010048
[4] N. Jardine and Cornelis Joost van Rijsbergen. 1971. The use of hierarchic clustering in information retrieval. *Inf. Storage Retr.* 7, 5 (1971), 217–240. https://doi.org/10.1016/0020-0271(71)90051-9
[5] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. 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 2020, Online, November 16-20, 2020*, Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, 6769–6781. https://doi.org/10.18653/V1/2020.EMNLP-MAIN.550
[6] 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. https://doi.org/10.1145/3539618.3591715

[7] Jurek Leonhardt, Henrik Müller, Koustav Rudra, Megha Khosla, Abhijit Anand, and Avishek Anand. 2024. Efficient neural ranking using forward indexes and lightweight encoders. *ACM Transactions on Information Systems* 42, 5 (2024), 1–34.

[8] 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. https://doi.org/10.1145/3485447.3511955

[9] 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. https://doi.org/10.18653/V1/2021.REPL4NLP-1.17

[10] 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, Washington DC, USA, July 14-18, 2024*, Grace Hui Yang, Hongning Wang, Sam Han, Claudia Hauff, Guido Zuccon, and Yi Zhang (Eds.). ACM, 1411–1419. https://doi.org/10.1145/3626772.3657856

[11] Sean MacAvaney, Nicola Tonellotto, and Craig Macdonald. 2022. Adaptive Re-Ranking as an Information-Seeking Agent. In *Proceedings of the CIKM 2022 Workshops co-located with 31st ACM International Conference on Information and Knowledge Management (CIKM 2022), Atlanta, USA, October 17-21, 2022 (CEUR Workshop Proceedings, Vol. 3318)*, Georgios Drakopoulos and Eleanna Kafeza (Eds.). CEUR-WS.org. https://ceur-ws.org/Vol-3318/paper9.pdf

[12] 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. https://doi.org/10.1145/3511808.3557231

[13] Craig Macdonald, Nicola Tonellotto, Sean MacAvaney, and Iadh Ounis. 2021. PyTerrier: Declarative Experimentation in Python from BM25 to Dense Retrieval. In *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, Gianluca Demartini, Guido Zuccon, J. Shane Culpepper, Zi Huang, and Hanghang Tong (Eds.). ACM, 4526–4533. https://doi.org/10.1145/3459637.3482013

[14] 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. https://doi.org/10.1109/TPAMI.2018.2889473

[15] Antonio Mallia, Michal Siedlaczek, Joel M. Mackenzie, and Torsten Suel. 2019. 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 (CEUR Workshop Proceedings, Vol. 2409)*, Ryan Clancy, Nicola Ferro, Claudia Hauff, Jimmy Lin, Tetsuya Sakai, and Ze Zhong Wu (Eds.). CEUR-WS.org, 50–56. https://ceur-ws.org/Vol-2409/docker08.pdf

[16] 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. https://doi.org/10.1145/1148170.1148246

[17] Thong Nguyen, Sean MacAvaney, and Andrew Yates. 2023. A Unified Framework for Learned Sparse Retrieval. In *Advances in Information Retrieval - 45th European Conference on Information Retrieval, ECIR 2023, Dublin, Ireland, April 2-6, 2023, Proceedings, Part III (Lecture Notes in Computer Science, Vol. 13982)*, Jaap Kamps, Lorraine Goeuriot, Fabio Crestani, Maria Maistro, Hideo Joho, Brian Davis, Cathal Gurrin, Udo Kruschwitz, and Annalina Caputo (Eds.). Springer, 101–116. https://doi.org/10.1007/978-3-031-28241-6_7

[18] 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

[19] Rodrigo Frassetto Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *CoRR* abs/1901.04085 (2019). arXiv:1901.04085 http://arxiv.org/abs/1901.04085

[20] 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. https://doi.org/10.18653/V1/2020.FINDINGS-EMNLP.63

[21] 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).

[22] 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).

[23] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* 21 (2020), 140:1–140:67. http://jmlr.org/papers/v21/20-074.html

[24] Stephen E Robertson. 1977. The probability ranking principle in IR. *Journal of documentation* 33, 4 (1977), 294–304.

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

[26] Howard R. Turtle and James Flood. 1995. Query Evaluation: Strategies and Optimizations. *Inf. Process. Manag.* 31, 6 (1995), 831–850. https://doi.org/10.1016/0306-4573(95)00020-H

[27] Xiao Wang, Sean MacAvaney, Craig Macdonald, and Iadh Ounis. 2022. An Inspection of the Reproducibility and Replicability of TCT-ColBERT. In *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, Enrique Amigó, Pablo Castells, Julio Gonzalo, Ben Carterette, J. Shane Culpepper, and Gabriella Kazai (Eds.). ACM, 2790–2800. https://doi.org/10.1145/3477495.3531721

[28] Qiang Wu, Christopher J. C. Burges, Krysta M. Svore, and Jianfeng Gao. 2010. Adapting boosting for information retrieval measures. *Inf. Retr.* 13, 3 (2010), 254–270. https://doi.org/10.1007/S10791-009-9112-1

[29] Yingrui Yang, Parker Carlson, Shanxiu He, Yifan Qiao, and Tao Yang. 2024. Cluster-based Partial Dense Retrieval Fused with Sparse Text Retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, Grace Hui Yang, Hongning Wang, Sam Han, Claudia Hauff, Guido Zuccon, and Yi Zhang (Eds.). ACM, 2327–2331. https://doi.org/10.1145/3626772.3657972

[30] Justin Zobel and Alistair Moffat. 2006. Inverted files for text search engines. *ACM Comput. Surv.* 38, 2 (2006), 6. https://doi.org/10.1145/1132956.1132959