Adaptive Re-Ranking

Ata Cinar Genc
University of Massachusetts Amherst
Amherst, MA, USA
agenc@umass.edu

Emir Kaan Korukluoglu
University of Massachusetts Amherst
Amherst, MA, USA
ekorukluoglu@umass.edu

James Allan
University of Massachusetts Amherst
Amherst, MA, USA
allan@cs.umass.edu

Abstract

Modern Information Retrieval (IR) systems typically use a "retrieve-then-rerank" pipeline, where a computationally expensive, predetermined cross-encoder re-ranks the top results from a fast initial retriever. While effective, this approach often applies heavy re-ranking models regardless of query complexity, resulting in high latency and wasted computational resources on simple queries. We propose Adaptive Re-Ranking, a framework that dynamically routes queries to the most cost-effective strategy—ranging from sparse retrieval (BM25) and dense re-ranking (MiniLM-L6-v2) to heavy neural re-ranking (BGE-v2-m3)—based on query complexity. To train our routing classifier, we introduce a novel utility function to label queries based on their retrieval effectiveness penalized by latency. Compared to BGE our method achieves 1.15 − 53x lower median latency and 1.11 − 3.52x lower mean latency across all datasets we have tested while delivering −19% to +5% effectiveness. Our findings show that routing queries based on our novel utility function offers a scalable solution for reducing computational costs and latency in a variety of IR systems.

ACM Reference Format:
Ata Cinar Genc, Emir Kaan Korukluoglu, and James Allan. 2026. Adaptive Re-Ranking. In *Proceedings of International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '26).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

The field of Information Retrieval (IR) has gone through a paradigm shift with the adoption of dense and cross-encoder neural networks. While traditional lexical models like BM25 [18] served as the cornerstone of retrieval systems for decades due to their unparalleled efficiency, lexical models often fail to capture the semantic meanings of complex queries. The introduction of the Transformer architecture [24] has also made it possible to use Transformers for ranking, using BERT to re-rank passages [15], using sentence-embeddings for similarity checks [17], and so on. More recently, powerful cross-encoder rerankers, such as **BGE-v2-m3** [3], have significantly improved retrieval accuracy. However, these models also have some drawbacks, such as an increase in latency up to 15x in our experiments, depending on the size of the collection.

Modern retrieval pipelines are structured as a two-stage process (Figure 1a): a fast, initial retriever followed by an expensive reranker. We argue that using the most effective, "heavy" cross-encoder for every single query might result in drastically high average latency, as well as enormous computational cost. On our test set, looking at the average of nDCG@10 and MRR@10, around 40% of the queries benefited from having a reranker and only 11% of the queries benefited from having a heavier reranker compared to a light reranker. This shows that a significant portion of user queries is simple enough to be accurately handled by a fast, low-cost method like BM25 or a lightweight model. Using a large model on every query constitutes a huge waste of computational resources.

To address this efficiency bottleneck, we propose **Adaptive Re-Ranking**—a framework that dynamically selects the optimal re-ranking strategy per query by predicting which model would perform optimally through a lightweight classifier. Our study introduces three key contributions:

(1) **A routing mechanism** that routes each query to one of three efficiency-aware pipelines:
   - **Class 0 (No Reranker):** Fastest execution with minimal computational using a lexical model.
   - **Class 1 (Light Reranker):** Balanced effectiveness and latency.
   - **Class 2 (Heavy Reranker):** Highest effectiveness at increased computational and latency cost.
(2) **A novel utility-based labeling** that defines optimal strategy selection through a trade-off between retrieval effectiveness (average of nDCG@10 and MRR@10) and a latency penalty, enabling dataset creation according to the desired latency penalty.
(3) **An empirical study** that shows adaptive routing can significantly reduce latency while maintaining competitive retrieval quality, validated across diverse datasets.

Our adaptive re-ranking framework offers significant practical value for resource-constrained development environments such as mobile edge computing (MEC). Recent work has [27] demonstrated that mobile devices face severe constraints in CPU, GPU, memory, and battery life. In such limited-resource environment, using heavy neural re-ranking models for all queries becomes impractical and wasteful. By dynamically routing queries, our classifier enables IR systems to achieve competitive retrieval quality while respecting computational and energy limitations. By adjusting λ in our utility function, speed and efficiency can be prioritized for such environments. Our code and data is available at https://anonymous.4open.science/r/Adaptive_Re-Rank-F354

2 Related Work

Query Performance Prediction. QPP's goal is to estimate retrieval effectiveness without any relevance judgments. Traditional QPP methods rely on clarity-based [5], robustness-based [28], and score-based [20] approaches that uses corpus statistics and query term frequencies as signals. Recent advances have introduced neural approaches to QPP such as an end-to-end neural framework [26]. The introduction of pre-trained language models further advanced the developments by fine-tuning BERT for QPP, achieving state-of-the-art performance with significantly lower latency than previous neural predictors [1]. Recent work has also explored using LLMs

SIGIR '26,
2026. ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

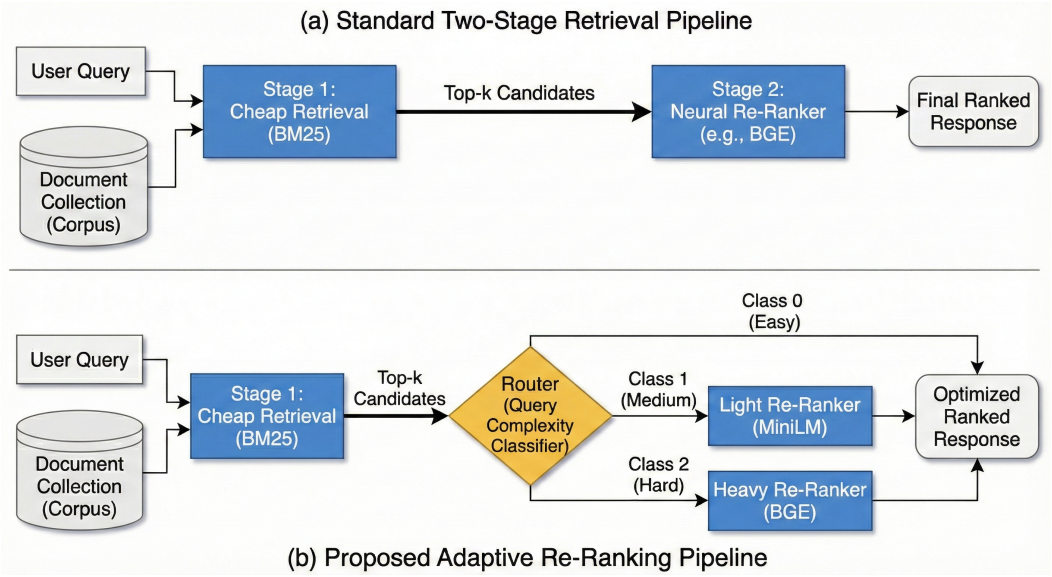

Figure 1: System Architecture Comparison. (a) The standard two-stage pipeline applies a heavy reranker to all queries. (b) Our Adaptive Re-Ranking pipeline uses a lightweight Router to decide the most cost-effective path (Class 0, 1, or 2) per query.

for QPP by, for example, fine-tuning open-source LLMs to generate relevance judgments to estimate effectiveness [14]. Despite these advances, prior work found that LLMs are not reliable enough to create relevance judgments [8]. Our work differs from traditional QPP: rather than predicting a performance score, we are routing a classification problem over re-ranking strategies, and we include latency into our novel utility-based labeling function.

Adaptive Retrieval. Adaptive retrieval methods use learned signals to choose between sparse, dense, and hybrid pipelines [2]. Other work claims that the first-stage ranker is the bottleneck in latency and proposes a unified prediction framework that chooses hyperparameters based on the given query [12]. More recently, a new paradigm has been explored that is similar to mixture of experts in structure and aims to leverage a zero-shot mixture of heterogeneous retrievers—including human-like sources—that dynamically weights and fuses them per query, yielding substantially better retrieval performance across diverse information needs [11]. Our work is similar in the way of offering a selection of retrievers, however we also consider the efficiency of the ideal reranker on a per-query basis.

Adaptive RAG Techniques. With the recent advances in Retrieval-Augmented Generation (RAG), new adaptive pipelines were introduced. Recent work employs a classifier to determine if RAG is necessary for a given prompt [10]. Similarly, MBA-RAG [19] uses reinforcement learning to optimize retrieval strategies. Other work proposes an adaptive router for different RAG paths (symbolic, neural, hybrid) [9]. Our work differs by focusing specifically on the re-ranking strategies in standard IR pipelines, introducing a supervised learning approach grounded in a cost-benefit (utility) function rather than relying on LLM-generated complexity signals.

3 Method

Our framework consists of a utility-based labeling that assigns each query to its optimal retrieval strategy, dataset curation, and router training, a routing pipeline that directs queries at inference time.

3.1 Labeling

To create our classifier, we label queries based on what retrieval method provides the best performance with a reasonable latency. For this, we define the effectiveness score as the average of nDCG@10 and MRR@10:

$$\text{eff}_m(q) = \frac{1}{2}\,\text{nDCG@10}_m(q) + \frac{1}{2}\,\text{MRR@10}_m(q). \tag{1}$$

This gives a value between $(0, 1)$ that captures how good the ranking is for that query, ignoring latency. While nDCG captures the overall ranking quality, MRR focuses on first relevant hit, which is quite important in modern information retrieval [13]. We combine both to provide the benefits of nDCG with the stronger emphasis on the first relevant item found.

However, we do not want to choose models only by effectiveness; slower models should be penalized. While recent works [2, 10, 19] explore the trade-off between retrieval accuracy and computational cost, they do not include measured true latency (clock time) in decision making. To be able to train our classifier with latency included we introduce a formula that penalizes the latency *per query* instead of taking the average of all queries. We are aiming to reward queries that only use more computational resources when doing so will actually improve that query itself.

Let $lat_m(q)$ be the latency of a given query of model m, and let $\max_j(\text{lat}_j(q))$ be the maximum latency among all models for that query. We mix the effectiveness and latency scores using λ to define

our utility function:

$$\text{util}_m(q) = (1-\lambda)\,\text{eff}_m(q) + \lambda\left(1 - \frac{\text{lat}_m(q)}{\max_j(\text{lat}_j(q))}\right). \quad (2)$$

Note that the model with the lowest latency will have a boosted utility score compared to the others; indeed, the most expensive model will receive no contribution to the utility from latency. Intuitively, $\text{util}_m(q)$ is high when the model is both effective and fast, and it decreases when the model is slow relative to the others.

The parameter λ serves as a hyperparameter controlling the system's sensitivity to latency. After conducting sensitivity analysis, we select $\lambda = 0.05$ to enforce a 'performance-first' policy[1]. This value ensures that while slower ranking methods (such as using BM25+BGE-v2-m3) incur a penalty, they are not systematically disqualified. Consequently, this formula implies that a significant increase in latency is acceptable only if it yields a measurable gain in retrieval effectiveness, effectively acting as a regularizer against inefficient models that offer negligible performance improvements. λ can be adjusted according to need: if the user desires the system to run faster, then it can be adjusted by increasing λ.

Label Category	Count	$\overline{\text{Eff}}_{\text{BM25}}$	$\overline{\text{Eff}}_{\text{L6}}$	$\overline{\text{Eff}}_{\text{BGE}}$
Class 0 (BM25)	182,876	0.2203	0.1801	0.1831
Class 1 (L6)	88,295	0.2049	0.7475	0.6467
Class 2 (BGE)	35,573	0.1614	0.3492	0.6441

Table 1: Label distribution and average effectiveness scores across the dataset, $\overline{X}$ denotes mean, calculated with Eq. (1)[2]

3.2 Dataset Curation

We curated a dataset using BEIR [23] datasets with 306,744 labeled queries with $\lambda = 0.05$. It shows a skew toward Class 0 (no reranking) of 59.6% (see Table 1) which indicates that the majority of queries can be handled efficiently with only BM25 without increasing re-ranking latency. However, queries labeled as Class 1 (MiniLM-L6) received higher average effectiveness, even outperforming the heavy BGE Model in that subset. It can also be observed that in Class 2 the average effectiveness scores of both BM25, and MiniLM-L6 is *significantly* lower than BGE, which illustrates that for some queries, a bigger model yields more effectiveness – but not always, which justifies our approach. Illustrating these patterns at the query level, Table 2 shows examples from each class and the utility scores across different retrieval models.

3.3 Retriever Decision

We choose `BM25` as the first-stage ranker based on its proven success and popularity. For light and heavy rerankers, we use `ms-marco-MiniLM-L6-v2` [25] and `bge-reranker-v2-m3` [3], again, based on their performance and popularity [6, 7]. We also explore other popular options such as `Qwen3-ReRanker-8B`, `Qwen3-ReRanker-0.6b` [21, 22] for the heavy reranker, and find that `bge-reranker-v2-m3` provides better accuracy and lower latency compared to the other models on the training data.

[1]We performed sensitivity analysis with the oracle across $\lambda \in \{0.01, 0.05, 0.1, 0.15, 0.2\}$. $\lambda = 0.05$ yielded the best balance, providing the least effectiveness decline with moderate latency reductions.
[2]L6 means BM25 + L6, BGE means BM25+BGE

Dataset	Query	Best Model	util BM25	util L6	util BGE
MS-Marco	is levaquin an antibiotic	bge	0.04	0.26	0.40
MS-Marco	what causes itching during exercise	l6	0.01	0.54	0.53
NF Corpus	What is 'Meat Glue'?	bm25	0.70	0.34	0.30
NF Corpus	The Top Three DNA Protecting Spices	bm25	0.63	0.62	0.58
Quora	How can I rule the world?	l6	0.29	0.98	0.95

Table 2: Example scores for BM25, MiniLM-L6, and BGE across MS-Marco, NFCorpus, and Quora queries ($\lambda = 0.05$)

3.4 Router Training

We train a supervised **query router** that predicts a routing class, given a query. Each class corresponds to a retrieval route (e.g., BM25 only, BM25 + Light Reranker, BM25 + Heavy Reranker). The routing goal is to select the most efficient route while maintaining ranking quality. Since our training data has imbalanced classes, we down-sample our dataset to match the minority class[3]. We use a 75/25 train-test ratio for training. We truncate queries and use fixed-length padding. Using this data, we fine tune `bert-base-uncased` with a sequence classification head `(num-labels=3)` using `AdamW` optimizer ($\eta = 3x10^{-5}$). We train the model for 20 epochs, and we select the best model based on validation accuracy. Final performance is reported as test accuracy of 65% with [0.649, 0.660] 95% confidence interval.

3.5 Experiment Setup

First Stage Retrieval We implement BM25 in Python with an in-memory inverted index using Porter/Snowball stemming and NLTK tokenization. Documents are indexed as *title* and *text* (when available). We use the same hardware for all ranking stages for consistency.

Candidate Set Size Throughout all experiments, we rerank the top $k = 50$ documents returned by the BM25 first-stage retriever.

Training and Testing We train the router on each dataset's official *train* split. All labelling is computed using the train split only, and test queries were only used on test experiments.

ReRankers We use the cross-encoder reranker *ms-marco-MiniLM-L-6-v2* (Sentence-Transformers `CrossEncoder`) to score each (q, d) pair in the top-50 candidate set and then sorted documents by the predicted relevance score to generate the ranked lists. We use `BAAI/bge-reranker-v2-m3` via FlagEmbedding (FP16 on GPU when available) to score the same (q, d) pairs and then sort documents by the reranker score.

4 Results

To measure the performance of our router against different methods, we conduct test runs in various IR datasets. For each query, we run the retrieval pipelines for each proposed method. We calculate the effectiveness (Eff) and latency scores per query. Table 3 evaluate the central claim of the paper: **Selective reranking is valuable because queries differ in how much they benefit from costly**

[3]We also experimented with class-weighted loss function instead of downsampling; however, it yielded lower routing accuracy than down-sampling.

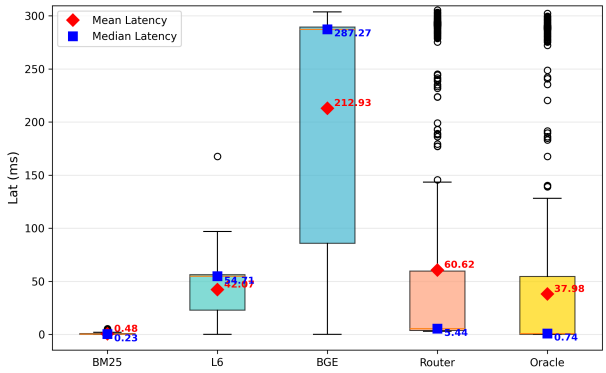

Figure 2: Latency distribution on NFCorpus

reranking. To isolate the potential of this proposed paradigm from the difficulty of learning, we report two oracle results: one at 100% accuracy (optimal routing), another one at 65% accuracy (showing imperfect routing) [4].

Selective reranking is a valid strategy across different datasets. Across all datasets, we see that fixed strategies always expose a latency-effectiveness tradeoff, whereas conditionally routing offers a balance. Looking at Table 3, the learned router achieved 1.15×–53.2× lower median latency and 1.11×–3.52× lower mean latency, with effectiveness changes ranging from −19% to +5%. The largest wins occurred on NFCorpus and SciFact, where median latency dropped by 53.2× and 33.6× (to 5.4 ms and 8.5 ms, respectively), while effectiveness slightly increased on NFCorpus (+4.7%) and decreased modestly on SciFact (−4.2%). The gap between mean and median reflects the handling of most queries through fast lexical retrieval, also shown in Figure 2[5]. In Quora, the router maintained almost identical effectiveness (−2.3%) while reducing the median latency by 35%. Compared to L6, the router's effectiveness remained close, with changes between −8.1% and +4.9%. At the same time, the router delivers median-latency gains on **NFCorpus** (10.1×) and **SciFact** (6.60×): the router lowers the latency distribution by routing to fast lexical paths, also seen in Figure 2.[5]

Oracle routing shows validity and potential generalizability. Across all datasets, Oracle shows the strongest performance in both effectiveness and latency, confirming that adaptive routing can dominate a fixed strategy (Table 3), establishing an upper bound which is unattainable by any fixed method – e.g., on **NFCorpus** Oracle (100%) shows 13% higher effectiveness against both **L6** and **BGE** whilst 1.10× and 5.60× faster, respectively. This shows that within each dataset, some queries benefit from costly routing and some do not.

Simulated oracle results show that the trained router is not random. The oracle at 65% accuracy shows how an imperfect router performs here. Looking at Table 3, we observe that the learned router's results are similar to the simulated results, showing us that the learned router does not behave like random guessing.

[4]This randomized inaccuracy also implies that the simulated router may sometimes choose an expensive route when a real, learned router would (incorrectly) choose a cheaper one, and vice versa.
[5]Since all datasets had similar results, we report only NFCorpus due to space constraints

Dataset	Metric	BM25	L6	BGE	Router	Sim(65%)	Oracle
FiQA	$\overline{Eff}$	0.27	0.37	0.42	0.34	0.40	0.46
	$\overline{Lat}$	19.9	71.2	299.4	128.5	109.7	90.5
	$\widetilde{Lat}$	18.2	70.1	298.7	69.1	61.8	32.2
Quora	$\overline{Eff}$	0.77	0.82	0.88	0.86	0.87	0.92
	$\overline{Lat}$	17.1	25.1	45.2	30.2	25.7	22.0
	$\widetilde{Lat}$	11.9	20.0	40.5	26.4	21.7	16.7
NFCorpus	$\overline{Eff}$	0.42	0.44	0.43	0.45	0.46	0.50
	$\overline{Lat}$	0.5	42.1	212.9	60.6	62.6	38.0
	$\widetilde{Lat}$	0.2	54.7	287.3	5.4	1.7	0.7
MS MARCO	$\overline{Eff}$	0.2	0.37	0.37	0.34	0.37	0.42
	$\overline{Lat}$	598	616	694	626	626	614
	$\widetilde{Lat}$	447	464	542	470	473	459
SciFact	$\overline{Eff}$	0.64	0.66	0.71	0.68	0.70	0.75
	$\overline{Lat}$	2.1	56.9	298.1	98.2	72.5	34.3
	$\widetilde{Lat}$	2.0	56.1	285.4	8.5	3.6	2.6

Table 3: Performance across all datasets ($\lambda = 0.05$). $\overline{X}$ denotes mean, $\widetilde{X}$ denotes median. Latency is reported in milliseconds.

Routing decision cannot be explained by lexical "complexity." To test whether the learned router is using the complexity of the words in a given query, we checked the queries against the Academic Word List [4] and CEFR-J advanced vocabulary (C1-C2) [16]. We observe that for all labeled queries, the percentage of advanced vocabulary remained consistent, showing that "advanced vocabulary" is not a selection criterion for the learned router (AWL's Cramer's $V = 0.03$, CEFR-J C1's $V = 0.04$, and C2's $V = 0.03$). We also observe that there are no significant differences across classes in query length (46.3-47.2 characters, $p = 0.30$) and word length (5.53-5.57 characters, $p = 0.41$), so length does not play a meaningful part in the routing decision.

5 Conclusion

In this paper, we propose an **Adaptive Re-Ranking** framework designed to mitigate the computational bottlenecks in modern two-stage retrieval systems. By formalizing the trade-off between retrieval effectiveness and system latency into a quantitative *utility* function, we developed a pipeline that dynamically routes queries to the most cost-effective reranking strategy, ranging from skipping re-ranking entirely (BM25) to a heavy cross-encoder model (BGE). Our experiments demonstrated that the "one-size-fits-all" paradigm, where one reranker is applied always, is computationally inefficient. We showed that a significant portion of queries benefit from lightweight models without compromising retrieval quality.

6 Limitations & Future Work

Domain scope Due to computational and time constraints, our training dataset might not capture a sufficiently representative variety of queries and domains, therefore our learned router may not be fully general. Future work should curate a diverse dataset that focuses on clean, high-quality multidomain datasets that enable models to learn multiple domain-specific signals, if there are any.
Model selection Our model selection was affected by the time and computational constraints. We also experimented with DistilBERT as a lighter router but observed a significant drop in accuracy, suggesting that our framework benefits from the full BERT capacity. Future work should explore other architectures and models.

Utility function We believe our utility function captures a good blend of effectiveness and latency of a selected model. Future work could refine the utility function, introduce new variables and more hyperparameters to adjust the weights according to personal needs.
Results interpretation Since our learned router is a black-box model, we cannot verify or explain the selection criteria that the learned router makes. Hence, we use an oracle with simulated 65% accuracy to interpret the results of our learned router, whether the results are a product of random noise or not.

Acknowledgements

This work was supported in part by the Center for Intelligent Information Retrieval (CIIR). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

[1] Negar Arabzadeh, Maryam Khodabakhsh, and Ebrahim Bagheri. 2021. BERT-QPP: Contextualized Pre-trained Transformers for Query Performance Prediction. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21)*. 2857–2861. doi:10.1145/3459637.3482063
[2] Negar Arabzadeh, Xinyi Yan, and Charles L. A. Clarke. 2021. Predicting Efficiency/Effectiveness Trade-offs for Dense vs. Sparse Retrieval Strategy Selection. arXiv:2109.10739 [cs.IR] https://arxiv.org/abs/2109.10739
[3] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. arXiv:2402.03216 [cs.CL]
[4] Averil Coxhead. 2000. A New Academic Word List. *TESOL Quarterly* 34, 2 (2000), 213–238. doi:10.2307/3587951
[5] Steve Cronen-Townsend, Yun Zhou, and W. Bruce Croft. 2002. Predicting Query Performance. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '02)*. 299–306. doi:10.1145/564376.564429
[6] Hugging Face. 2024. BAAI/bge-reranker-v2-m3 · Hugging Face. https://huggingface.co/BAAI/bge-reranker-v2-m3.
[7] Hugging Face. 2026. cross-encoder/ms-marco-MiniLM-L6-v2 · Hugging Face. https://huggingface.co/cross-encoder/ms-marco-MiniLM-L6-v2.
[8] Guglielmo Faggioli, Laura Dietz, Charles L. A. Clarke, Gianluca Demartini, Matthias Hagen, Claudia Hauff, Noriko Kando, Evangelos Kanoulas, Martin Potthast, Benno Stein, and Henning Wachsmuth. 2023. Perspectives on Large Language Models for Relevance Judgment. In *Proceedings of the 2023 ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR '23)* (Taipei, Taiwan). doi:10.1145/3578337.3605136
[9] Safayat Bin Hakim, Muhammad Adil, Alvaro Velasquez, and Houbing Herbert Song. 2025. SymRAG: Efficient Neuro-Symbolic Retrieval Through Adaptive Query Routing. In *Proceedings of The 19th International Conference on Neurosymbolic Learning and Reasoning (NeSy) (Proceedings of Machine Learning Research, Vol. 284)*. PMLR, 540–564. https://proceedings.mlr.press/v284/hakim25a.html
[10] Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. 2024. Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity. *arXiv preprint arXiv:2403.14403* (2024). https://arxiv.org/abs/2403.14403
[11] Jushaan Singh Kalra, Xinran Zhao, To Eun Kim, Fengyu Cai, Fernando Diaz, and Tongshuang Wu. 2025. MoR: Better Handling Diverse Queries with a Mixture of Sparse, Dense, and Human Retrievers. *arXiv preprint arXiv:2506.15862* (2025). https://arxiv.org/abs/2506.15862
[12] Joel Mackenzie, J. Shane Culpepper, Roi Blanco, Matt Crane, Charles L. A. Clarke, and Jimmy Lin. 2018. Query Driven Algorithm Selection in Early Stage Retrieval. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining* (Marina Del Rey, CA, USA) *(WSDM '18)*. Association for Computing Machinery, New York, NY, USA, 396–404. doi:10.1145/3159652.3159676
[13] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval*. Cambridge university press. https://nlp.stanford.edu/IR-book/
[14] Chuan Meng, Negar Arabzadeh, Arian Askari, Mohammad Aliannejadi, and Maarten de Rijke. 2024. Query Performance Prediction using Relevance Judgments Generated by Large Language Models. *arXiv preprint arXiv:2404.01012* (2024). https://arxiv.org/abs/2404.01012
[15] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *arXiv preprint arXiv:1901.04085* (2019). https://arxiv.org/abs/1901.04085
[16] Open Language Profiles. 2021. CEFR-J Wordlist for English. https://github.com/openlanguageprofiles/olp-en-cefrj.
[17] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 3982–3992. doi:10.18653/v1/D19-1410
[18] Stephen E Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR'94*. Springer, 232–241. doi:10.1007/978-1-4471-2099-5_24
[19] Xiaqiang Tang, Qiang Gao, Jian Li, Nan Du, Qi Li, and Sihong Xie. 2025. MBA-RAG: a Bandit Approach for Adaptive Retrieval-Augmented Generation through Question Complexity. In *Proceedings of the 31st International Conference on Computational Linguistics*, Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert (Eds.). Association for Computational Linguistics, Abu Dhabi, UAE, 3248–3254. https://aclanthology.org/2025.coling-main.218/
[20] Yongquan Tao and Shengli Wu. 2014. Query Performance Prediction By Considering Score Magnitude and Variance Together. In *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management (CIKM '14)*. 1891–1894. doi:10.1145/2661829.2661906
[21] Qwen Team. 2025. *Qwen3-Reranker-0.6B*. https://huggingface.co/Qwen/Qwen3-Reranker-0.6B
[22] Qwen Team. 2025. *Qwen3-Reranker-8B*. https://huggingface.co/Qwen/Qwen3-Reranker-8B
[23] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models. *arXiv preprint arXiv:2104.08663* (4 2021). https://arxiv.org/abs/2104.08663
[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, Vol. 30. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html
[25] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 33. 5776–5788. https://proceedings.neurips.cc/paper/2020/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html
[26] Hamed Zamani, W. Bruce Croft, and J. Shane Culpepper. 2018. Neural Query Performance Prediction using Weak Supervision from Multiple Signals. In *Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '18)*. 105–114. doi:10.1145/3209978.3210041
[27] Xiaojie Zhang and Saptarshi Debroy. 2023. Resource Management in Mobile Edge Computing: A Comprehensive Survey. *Comput. Surveys* 55, 13s (2023), 1–37. doi:10.1145/3589639
[28] Yun Zhou and W. Bruce Croft. 2007. Query Performance Prediction in Web Search Environments. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '07)*. 543–550. doi:10.1145/1277741.1277835