Evaluation of Intermediate Steps in Retrieval Augmented Generation

ARAM AYTEKIN, Universität Kassel, Germany
MORITZ DIETRICH, Universität Kassel, Germany
MAXIMILIAN BENNEDIK, Universität Kassel, Germany
DOMENIC BREMMER, Universität Kassel, Germany

This work investigates the research question: Does a RAG pipeline that expands the initial prompt into an LLM-generated query pool produce answers that humans prefer (on correctness, conciseness, and relevance) over a baseline that directly queries Elasticsearch once? To explore this, we implement two retrieval-augmented generation (RAG) pipelines that operate exclusively on search engine result page (SERP) snippets, which are often incomplete and inconsistent. The baseline pipeline (P1) retrieves snippets from a single query and directly conditions an LLM on this evidence. The advanced pipeline (P4), in contrast, employs LLM-based query expansion to generate a pool of reformulated queries, filters the resulting snippets, and integrates them for answer inference. Human evaluation indicates that P4 produces answers that are slightly more coherent, contextually appropriate, and preferred overall, though the improvements remain moderate. However, these gains come at the cost of significantly longer execution time, highlighting a trade-off between answer quality and system efficiency. These findings demonstrate the potential of query-expansion-based RAG pipelines for enhancing answer quality from fragmented snippet collections, while also pointing to open challenges in efficiency, evidence integration, and domain adaptation.

CCS Concepts: • **Information systems** → **Retrieval models and ranking**; **Evaluation of retrieval results**; • **Computing methodologies** → *Natural language processing*.

Additional Key Words and Phrases: RAG, Summary, Pooling, Query, LLM, Pipeline

ACM Reference Format:
Aram Aytekin, Moritz Dietrich, Maximilian Bennedik, and Domenic Bremmer. 2025. Evaluation of Intermediate Steps in Retrieval Augmented Generation. 1, 1 (August 2025), 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Large language models (LLMs) and retrieval-augmented generation (RAG) have transformed information retrieval and answer generation. Yet, in many real-world scenarios such as search engines, only fragmented snippets from search engine result pages (SERPs) are available rather than full documents. These snippets are incomplete and heterogeneous, making coherent answer generation challenging but practically important.

A key issue lies in query handling. Single-query retrieval often misses relevant evidence, while LLM-driven query expansion promises broader coverage and more diverse evidence. Whether such query pooling improves answer quality from SERP snippets remains unclear. Our study therefore asks: *Does a RAG pipeline with LLM-based query expansion produce answers that humans prefer—on correctness, conciseness, and relevance—over a baseline with a single direct query?*

To address this, we compare two pipelines operating solely on SERP snippets. The baseline (P1) issues one query and generates answers directly. The advanced pipeline (P4) expands the query into a pool, retrieves and filters additional snippets, and conditions generation on this evidence. We evaluate both through human preference judgments and

Authors' Contact Information: Aram Aytekin, Universität Kassel, Germany, uk097201@student.uni-kassel.de; Moritz Dietrich, Universität Kassel, Germany, uk097299@student.uni-kassel.de; Maximilian Bennedik, Universität Kassel, Germany, uk095879@student.uni-kassel.de; Domenic Bremmer, Universität Kassel, Germany, uk095482@student.uni-kassel.de.

2025. Manuscript submitted to ACM

runtime cost measurements. Results show that P4 yields slightly more coherent answers but at substantially higher cost, revealing a trade-off between quality and efficiency in query-expansion-based RAG pipelines.

2 Background & Related Work

2.1 Retrieval Augmented Generation

Retrieval-Augmented Generation (RAG) combines LLMs with external retrieval to produce factually grounded answers [6, 8]. By conditioning responses on retrieved documents, RAG mitigates hallucinations and improves factuality in tasks such as open-domain QA and summarization [5, 7]. Unlike most prior work assuming access to full documents, our study targets fragmented SERP snippets, which introduces challenges for coherent answer generation.

2.2 Query Expansion vs. Single-Query Baselines

Single-query retrieval is efficient but risks missing evidence. Query expansion addresses this by generating variant queries, improving coverage but also adding cost and potential noise [10, 15]. Our baseline (P1) represents the single-query setup, while P4 applies LLM-driven expansion within RAG, enabling systematic cost–quality comparison.

2.3 Human Preference Evaluation

Automatic metrics such as BLEU [12], ROUGE [9], or BERTScore [18] often fail to capture open-ended answer quality. Human evaluation remains the gold standard, typically via pairwise preference judgments [3, 14]. Crowdsourcing offers scalable evaluation for RAG systems [4]. In our study, we compare P1 vs. P4 answers along correctness, conciseness, and relevance, providing direct evidence for the impact of query pooling on perceived quality.

3 Methods

3.1 Pipelines

In the project [2], we evaluated four retrieval-augmented generation (RAG) pipelines that employ different strategies for query formulation and execution prior to generating LLM-based responses. While all four pipelines were initially tested, Pipeline 2 (LLM Summary before answer) and Pipeline 3 (LLM reformulates elasticsearch query) demonstrated marginal or negligible performance improvements compared to Pipeline 1 and Pipeline 4, respectively. Consequently, this study focuses exclusively on the comparative analysis between Pipeline 1 and Pipeline 4, which represent the most distinct and effective approaches among the evaluated systems.

Pipeline 1: Direct Retrieval and Answering. In this baseline approach, the system directly takes the user's input prompt and submits it as a query to the Elasticsearch retriever. The retriever returns the top-k relevant text snippets, which are then provided to the LLM together with a strict system instruction: the model must formulate its answer *only based on the retrieved snippets*. This ensures that the output is grounded in the search results without additional query expansion or prompt reformulation.

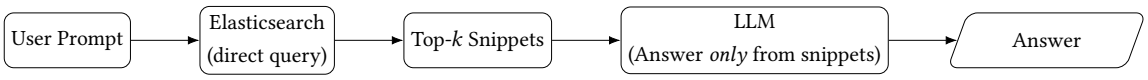

Fig. 1. Pipeline 1 — direct retrieval from Elasticsearch and grounded answering from retrieved snippets.

Pipeline 4: Query Pool Expansion. This enhanced approach adds an intermediate reasoning step before retrieval. The LLM is first asked to generate a *query pool*, i.e. a set of reformulated or semantically related search queries derived from the original user prompt. These multiple queries are then executed against Elasticsearch, leading to a richer and more diverse set of retrieved snippets. The snippets are aggregated and passed to the LLM, which produces the final answer. By broadening the retrieval space, Pipeline 4 aims to increase coverage and relevance compared to the single-query baseline.

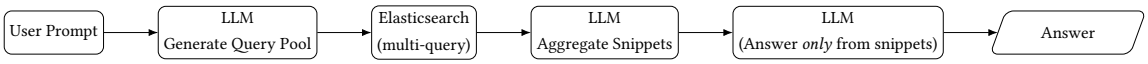

Fig. 2. Pipeline 4 — LLM-generated query pool, multi-query retrieval, snippet aggregation, and grounded answering.

3.2 Retrieval configuration

Data. We retrieve over an Elasticsearch index built from the *Archive Query Log (AQL-22)* [13], a corpus of archived search engine result pages (SERPs) containing historical queries and their snippet-level results (titles, snippets, target URLs). Crucially, because these SERPs are captured from the Internet Archive, the queries and their retrieved contexts are *reproducible*; unlike commercial engines (e.g., Google) whose indices and ranking algorithms change over time, AQL provides a stable snapshot that makes our comparative analysis possible.

Source retriever. Our retriever queries the AQL-backed Elasticsearch index to collect snippet contexts that ground LLM answers. In **Pipeline 1**, the user prompt is issued as a single ES query; we take the top-k snippets and pass them to the LLM with an instruction to answer *only* from those snippets. In **Pipeline 4**, the LLM first generates a pool of reformulated queries; we execute each against the same index, merge and deduplicate the returned snippets, then feed the aggregated context to the LLM. Both pipelines operate on the identical index and use the same k and filtering/cleanup steps to isolate the effect of the query strategy itself.

3.3 LLM Configuration

This subsection describes the configuration and prompting strategy used for the large language models (LLMs) in our pipelines. Each LLM instance operates with a specialized system prompt defining its role, objectives, and constraints. The general workflow across pipelines includes generating relevant queries, retrieving or aggregating context snippets, and producing answers grounded in those snippets.

All LLM calls in this work are performed using the Helmholtz-Blablador API [11] with the model `"alias-large"` [16], also referred to as Qwen3-30B-A3B, optimized for information retrieval and question-answering tasks.

The following sections detail the three main stages of LLM configuration: *Query Pool Generation*, *Snippet Aggregation*, and *Answer Generation*. Specific pipelines may implement subsets of these stages or additional mechanisms.

Query Pool Generation. The system prompt guides the LLM in producing concise and meaningful search queries from a user's question by extracting core subjects, typically the most important nouns, entities, or concepts. This ensures that generated queries are focused on essential terms for downstream retrieval.

Snippet Aggregation. After generating a query pool and retrieving candidate snippets, the LLM evaluates which snippets are relevant for answering the user's question. The model is instructed to retain any snippet that might

provide useful information, including direct answers, relevant facts, supporting examples, or background context, and to exclude only unrelated content.

Answer Generation. In both Pipelines 1 and 4, the final step is to generate a concise answer strictly based on the selected context snippets. The system prompt instructs the LLM to provide clear, direct responses in no more than five sentences. Only information present in the snippets may be used; if the context lacks sufficient information, the model indicates that no answer is available.

3.4 Evaluation Protocol

We curated a set of 20 predefined user questions and executed each question on both systems: **Pipeline 1** (single direct query) and **Pipeline 4** (LLM-generated query pool). For each prompt, this produced two answers, one from each pipeline, generated under the same model and decoding settings.

Human evaluation was conducted as a randomized, blind A/B test. For every prompt, the two answers were shown side-by-side with left/right positions randomized and the system identities hidden. Annotators followed the instructions in Section 3.5 (correctness as the primary criterion, then relevance and conciseness) and selected the preferred answer; a "No preference" option was available for ties.

Our primary metric is the *win rate*: the proportion of pairwise comparisons in which a pipeline's answer was preferred, computed over non-tied votes. We report win rates per pipeline to compare overall preference.

3.5 Annotator Remarks

Annotators compared two anonymous answers for each prompt and selected the better one according to three criteria. The primary dimension was **Correctness**, i.e., factual accuracy, avoidance of hallucinations, and consistency with the question. Secondary criteria were:

- **Relevance**: whether the answer stays on-topic and addresses the prompt,
- **Conciseness**: whether the response is complete yet compact, without unnecessary repetition.

The evaluation followed a strict priority order: **Correctness** → **Relevance** → **Conciseness**. When criteria conflicted, annotators were instructed to prefer the answer that was more accurate and cautious, even if it was less polished. If both answers were equally good (or equally poor), annotators could indicate *No preference*.

After each vote, annotators briefly justified their decision with a short comment referring to the criteria (e.g., correctness of facts, conciseness, or topicality). Critical errors or omissions were expected to be highlighted in these remarks.

Finally, annotators were reminded to judge only the written content itself: unsupported or incorrect claims were to be penalized more heavily than cautious ones, and external searches were not permitted. Stylistic polish or formatting was not to influence voting unless it directly affected clarity.

3.6 Hypotheses

Rationale. Pipeline 4 expands the initial prompt into an LLM-generated *query pool*, increasing search coverage and expected recall of relevant snippets. Greater coverage should yield answers that are more accurate and on-topic, which, under our evaluation protocol, should translate into higher preference.

Primary hypothesis. $\mathbf{H_1}$: Pipeline 4's win rate exceeds 50% over non-tied pairwise comparisons with Pipeline 1. $\mathbf{H_0}$: Pipeline 4's win rate is $\leq 50\%$.

Note on latency. Pipeline 4 may incur higher latency due to multi-query retrieval and increased LLM Overhead. We report latency distributions and mean times in the Results section, but latency is not part of the primary hypothesis test.

3.7 Cost Measurement

What we measure. To assess our secondary latency hypothesis (Section 3.6), we record the *LLM runtime* per prompt and pipeline: the **sum of time passed during all LLM calls** involved in producing an answer. For **Pipeline 1** this is the single answer-generation call. For **Pipeline 4** this includes the query-pool *expansion* call, the filtering and aggregation call, and the final *answer-generation* call. Measurements are taken at the application level and therefore include model inference time plus client–server overhead for the LLM service.

Scope and aggregation. We do not include Elasticsearch retrieval time or disk I/O—only LLM-invocation time. For each of the 20 questions and each pipeline, we compute the mean runtime across runs; we also report overall means and show per-question comparisons in the timing figures.

Cost perspective. Pipeline 1 is the lower bound, issuing exactly one LLM call. Pipeline 4 typically makes three (query expansion, snippet filtering/summarization, final answer), which raises latency and energy use. The filtering stage is the heaviest, since it ingests the largest raw snippet set; however, it often *shrinks* the final context, partially offsetting token cost in the last generation step. In short: Pipeline 4 pays extra up front to reduce the payload of the final call.

Given that our model (`Qwen3-30B-A3B`) is an MoE (128 experts, 8 active), per-token compute is lower than dense models; prior work reports roughly 30–50% savings, which helps amortize Pipeline 4's additional calls [16, 17].

Limitations. Runtimes are not normalized by token counts and will co-vary with prompt/context length (Pipeline 4 typically supplies longer contexts due to multi-query aggregation). Transient network/provider variability may affect measurements. Because non-LLM retrieval stages are excluded, these numbers reflect the LLM portion of end-to-end latency and likely *underestimate* total wall-clock differences between pipelines.

4 Results

The win-rate figure (Fig. 3) aggregates all pairwise votes across 20 prompts [1], which were carefully curated to span diverse domains (e.g., scientific topics, general knowledge) and balanced across difficulty levels. It shows a clear advantage for **Pipeline 4** (43.8%) over **Pipeline 1** (25.0%), with a substantial proportion of evaluations (31.2%) resulting in tied outcomes where annotators could not distinguish between the pipeline responses. As defined in Section 3.4, "win rate" denotes the fraction of all comparisons in which a pipeline's answer was preferred under our annotator guidelines (Section 3.5), while "Don't Care" represents cases where both responses were deemed equivalent in quality. In total, evaluations were conducted by **12 independent annotators**. The high frequency of tied evaluations suggests that while Pipeline 4 demonstrates superior performance when differences are detectable, both pipelines often produce comparable quality responses, indicating that the performance gap, though consistent, may be context-dependent.

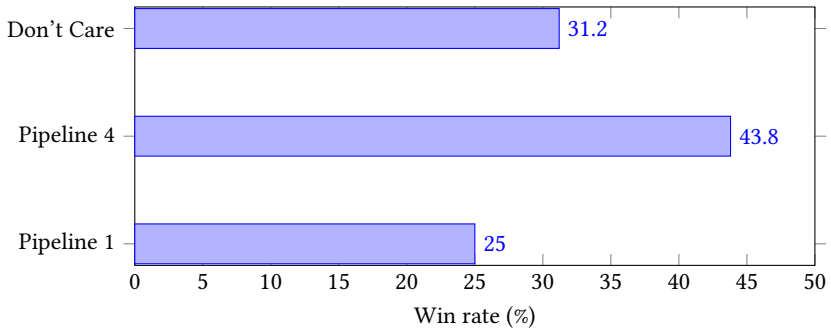

Fig. 3. Win rate comparison between Pipeline 1, Pipeline 4, and Don't Care outcomes. Data from Voting [1]

Performance comparison. Pipeline 4 achieves approximately 57% higher win rate than Pipeline 1, as shown in Fig. 3, indicating a significant and consistent preference in randomized, blind human voting. The additional query-pool expansion steps also lead to roughly four times longer runtime per question (Fig. 4), highlighting a clear cost–quality trade-off.

Retrieval coverage limits. Not infrequently, Elasticsearch retrieved *no accurate, on-topic snippets* for a given prompt. As documented in our evaluation set [1], for example, the question "How does photosynthesis work?", both pipelines struggled to obtain relevant SERP evidence, leading to the LLM being unable to answer. In such cases, the attainable performance of either pipeline is compressed, dampening any advantage from Pipeline 4's broader query coverage.

Knowledge override on sensitive topics. We observed instances where the model answered *regardless* of available context, i.e., overrides content based on own knowledge, particularly for highly sensitive topics (e.g., COVID-19, climate change). As documented in our evaluation set [1], the question "How does the greenhouse effect work?", both pipelines produced detailed answers with little to no support in the retrieved snippets, violating the "answer only from snippets" instruction and directly harming the **Correctness** criterion. This introduces variance in human preference that can overshadow small win-rate differences.

Cost–latency perspective. Pipeline 4's additional LLM calls for query-pool expansion (and occasional aggregation) increase runtime and token usage roughly linearly with the extra steps. Given the marginal improvement in win rate, the benefit may be questionable when resources or latency budgets are tight.

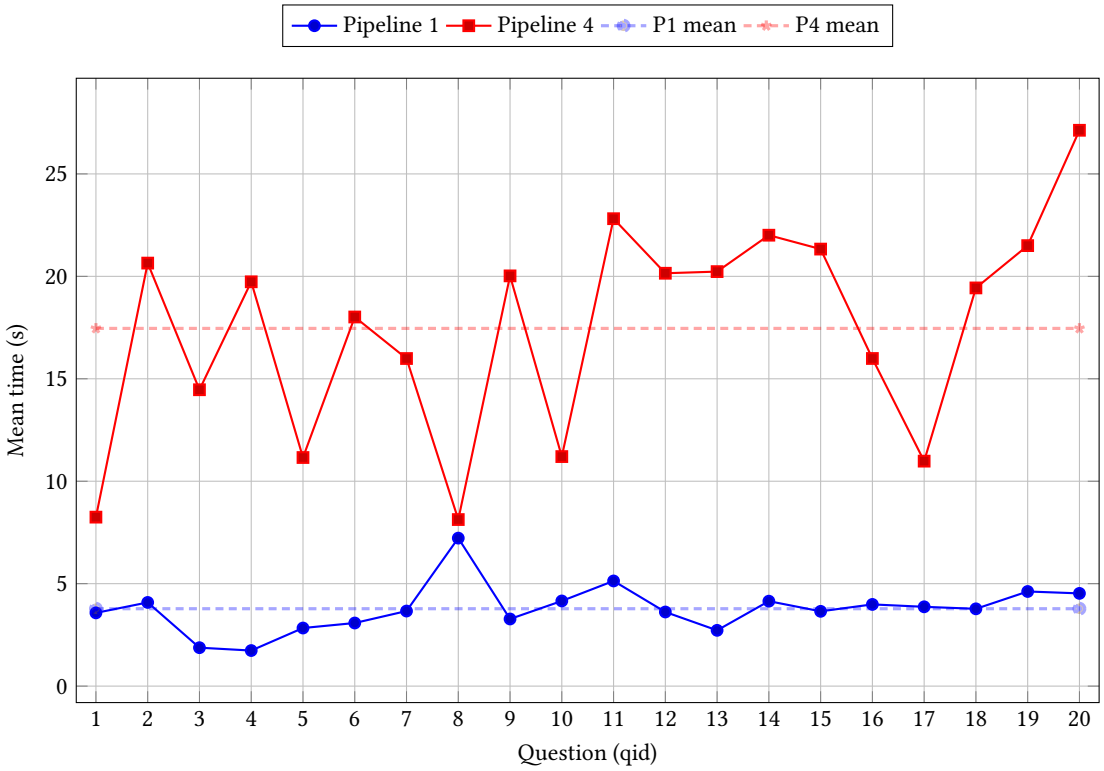

Fig. 4. Mean LLM response times per question (s) for the two pipelines.

Interpreting downward spikes. In Fig. 4, the sharp *downward* spikes for Pipeline 4 coincide with cases where Elasticsearch failed to retrieve meaningful snippets. With little or no usable context, the expansion step yields minimal material and the final answer call is short (often an abstention or terse reply), reducing measured LLM runtime. These dips therefore reflect a *retrieval failure*, not an efficiency gain of the multi-query pipeline.

5 Conclusion

We compared two RAG pipelines on 20 predefined prompts over the reproducible Archive Query Log (AQL) index: a single-query baseline (Pipeline 1) and a query-pool variant (Pipeline 4).

Rationale. Pipeline 4 expands the initial prompt into an LLM-generated *query pool*, increasing search coverage and expected recall of relevant snippets. Greater coverage should yield answers that are more accurate and on-topic, which, under our evaluation protocol, should translate into higher preference.

Limitations. Three factors temper this advantage:
(i) retrieval coverage gaps where Elasticsearch returned no meaningful snippets
(ii) occasional knowledge overrides—especially on sensitive topics—where the model answered without sufficient grounding

(iii) higher LLM runtime for Pipeline 4 due to additional expansion/aggregation steps. Timing analysis shows that apparent "fast" spikes for Pipeline 4 coincide with poor retrieval (short contexts), not genuine efficiency gains.

Overall. Despite the limitations, query-pool expansion provides consistent benefits in well-grounded settings. Our results support H_1 and show that Pipeline 4 has a modest advantage, though at the cost of increased runtime. In resource-constrained environments, this cost–quality trade-off requires careful consideration.

References

[1] Aram Aytekin, Domenic Bremmer, Moritz Dietrich, and Maximilian Bennedik. 2025. Evaluation SLT Project. https://github.com/CuzImAram/slt_project/tree/main/eval

[2] Aram Aytekin, Domenic Bremmer, Moritz Dietrich, and Maximilian Bennedik. 2025. SLT Project. https://github.com/CuzImAram/slt_project

[3] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. 2022. Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. arXiv:2204.05862 [cs.CL] https://arxiv.org/abs/2204.05862

[4] Lukas Gienapp, Tim Hagen, Maik Fröbe, Matthias Hagen, Benno Stein, Martin Potthast, and Harrisen Scells. 2025. The Viability of Crowdsourcing for RAG Evaluation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '25)*. ACM, 159–169. https://doi.org/10.1145/3726302.3730093

[5] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: Retrieval-Augmented Language Model Pre-Training. arXiv:2002.08909 [cs.CL] https://arxiv.org/abs/2002.08909

[6] Gautier Izacard and Edouard Grave. 2021. Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty (Eds.). Association for Computational Linguistics, Online, 874–880. https://doi.org/10.18653/v1/2021.eacl-main.74

[7] 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. https://doi.org/10.18653/v1/2020.emnlp-main.550

[8] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 9459–9474. https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf

[9] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*. Association for Computational Linguistics, Barcelona, Spain, 74–81. https://aclanthology.org/W04-1013/

[10] Rodrigo Nogueira and Kyunghyun Cho. 2020. Passage Re-ranking with BERT. arXiv:1901.04085 [cs.IR] https://arxiv.org/abs/1901.04085

[11] Helmholtz AI Platform. [n. d.]. Blablador API Access Guide. https://sdlaml.pages.jsc.fz-juelich.de/ai/guides/blablador_api_access/#step-2-obtain-an-api-key

[12] Matt Post. 2018. A Call for Clarity in Reporting BLEU Scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana Neves, Matt Post, Lucia Specia, Marco Turchi, and Karin Verspoor (Eds.). Association for Computational Linguistics, Brussels, Belgium, 186–191. https://doi.org/10.18653/v1/W18-6319

[13] Jan Heinrich Reimer, Sebastian Schmidt, Maik Fröbe, Lukas Gienapp, Harrisen Scells, Benno Stein, Matthias Hagen, and Martin Potthast. 2023. The Archive Query Log: Mining Millions of Search Result Pages of Hundreds of Search Engines from 25 Years of Web Archives. In *46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2023)*, Hsin-Hsi Chen, Wei-Jou (Edward) Duh, Hen-Hsen Huang, Makoto P. Kato, Josiane Mothe, and Barbara Poblete (Eds.). ACM, 2848–2860. https://doi.org/10.1145/3539618.3591890

[14] Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. 2022. Learning to summarize from human feedback. arXiv:2009.01325 [cs.CL] https://arxiv.org/abs/2009.01325

[15] Liang Wang, Nan Yang, and Furu Wei. 2023. Query2doc: Query Expansion with Large Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 9414–9423. https://doi.org/10.18653/v1/2023.emnlp-main.585

[16] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang,

Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. 2025. Qwen3 Technical Report. arXiv:2505.09388 [cs.CL] https://arxiv.org/abs/2505.09388

[17] Longfei Yun, Yonghao Zhuang, Yao Fu, Eric P Xing, and Hao Zhang. 2024. Toward Inference-optimal Mixture-of-Expert Large Language Models. arXiv:2404.02852 [cs.LG] https://arxiv.org/abs/2404.02852

[18] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating Text Generation with BERT. arXiv:1904.09675 [cs.CL] https://arxiv.org/abs/1904.09675