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

Information retrieval and answer generation have been transformed in recent years by large language models (LLMs) and retrieval-augmented generation (RAG). While these approaches typically rely on access to the full underlying documents, many real-world scenarios such as search engines only provide access to fragmented snippets contained in search engine result pages (SERPs). These snippets are often incomplete, inconsistent, and heterogeneous in style, making coherent answer generation particularly challenging. Nevertheless, being able to synthesize high-quality answers from such fragments is of both practical and theoretical interest.

A central obstacle in this setting lies in how queries are handled. Traditional retrieval methods issue a single query and directly return associated results, which often fails to capture the full breadth of relevant information. Recent advances in LLM prompting, however, allow for automatic query reformulation and expansion, potentially improving coverage and evidence diversity. Whether such query pooling strategies can indeed enhance answer generation quality from SERP snippets remains an open question.

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

1.1 Research Question

Against this background, our study investigates the following 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?*

1.2 Contribution

To answer this question, we implement and evaluate two RAG pipelines that operate solely on SERP snippets. The baseline pipeline (P1) issues a single query and generates answers from the retrieved snippets without further reformulation. The advanced pipeline (P4), in contrast, expands the initial query into a pool of reformulated queries using an LLM, retrieves and filters a broader snippet set, and conditions answer generation on this richer evidence. We evaluate both pipelines through human preference judgments along correctness, conciseness, and relevance, and further measure computational cost. Our findings show that while P4 produces answers that are slightly more coherent and preferred overall, it incurs substantially higher runtime costs, highlighting a trade-off between answer quality and efficiency. This contribution provides insight into the potential and the limitations of query-expansion-based RAG pipelines when applied to fragmented snippet data.

2 Background & Related Work

2.1 Retrieval Augmented Generation

Retrieval-Augmented Generation (RAG) has emerged as a powerful paradigm for combining the generative capabilities of large language models (LLMs) with external retrieval mechanisms to produce factually grounded answers. In typical RAG setups, queries are used to retrieve relevant documents or passages from a large corpus, which are then used to condition the LLM's response [4, 7]. This approach mitigates hallucinations often observed in standalone LLMs and enhances factual accuracy. Previous studies have demonstrated the effectiveness of RAG in settings such as open-domain question answering, summarization, and knowledge-intensive tasks [3, 5]. However, most prior work assumes access to complete and well-structured documents. In contrast, the present study operates under the more constrained setting of search engine result page (SERP) snippets, which are often fragmented, inconsistent, and incomplete. This constraint introduces unique challenges for coherent answer generation, motivating the exploration of advanced retrieval strategies.

2.2 Query Expansion vs. Single-Query Baselines

Traditional information retrieval systems often rely on a single-query approach, issuing one query to retrieve the top-k documents or passages. While computationally efficient, this strategy may fail to capture the full spectrum of relevant evidence, particularly in domains with heterogeneous or incomplete sources. Query expansion techniques aim to address this limitation by reformulating the initial query or generating multiple variant queries to improve coverage. Classical methods include pseudo-relevance feedback and relevance-based models, whereas more recent approaches leverage LLMs to generate paraphrases, sub-questions, or expanded queries tailored to the information need [9, 15]. The trade-off is evident: query expansion can increase evidence diversity and retrieval effectiveness, but at the cost of additional computation and potential introduction of irrelevant information. In our study, the baseline pipeline (P1) represents the single-query approach, while the advanced pipeline (P4) operationalizes LLM-driven query expansion within a RAG framework, allowing for a systematic evaluation of this trade-off.

2.3 Human Preference Evaluation

Automatic evaluation metrics such as BLEU [11], ROUGE [8], and BERTScore [17] are limited in their ability to capture the quality of open-ended answers, especially in terms of correctness, conciseness, and relevance [5]. Human evaluation remains the gold standard for assessing answer quality in generative tasks [1, 14]. Pairwise preference judgments, where annotators compare outputs from different systems, have been widely adopted to obtain robust insights into model performance [13].

Recent work has shown that crowdsourcing can be a viable approach for evaluating RAG systems, enabling structured assessments while controlling for annotator variability [2]. Despite the advantages, human evaluation remains resource-intensive. In our study, we conduct a structured human preference evaluation comparing answers generated by P1 and P4 along three dimensions: correctness, conciseness, and relevance. 3.5 This provides direct evidence of the practical impact of query pooling on perceived answer quality, complementing quantitative retrieval metrics and highlighting the cost-quality trade-offs inherent in LLM-driven RAG pipelines.

3 Methods

3.1 Pipelines

In this project, we compare two retrieval-augmented generation pipelines that differ in how they formulate and execute search queries before producing an LLM-based answer.

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.

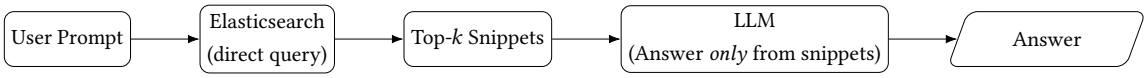

Fig. 1. Pipeline 1 — direct retrieval from Elasticsearch and grounded answering from retrieved snippets.

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.

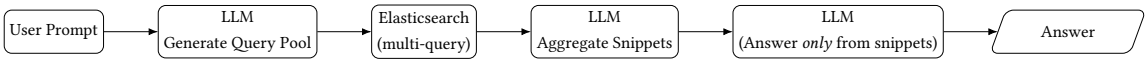

Fig. 2. Pipeline 4 — LLM-generated query pool, multi-query retrieval, snippet aggregation, and grounded answering.

3.2 Retrieval configuration

Data. We retrieve over an Elasticsearch index built from the *Archive Query Log (AQL-22)* [12], 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 [10] with the model `"alias-large"` [6], 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. From a computational cost perspective, Pipeline 1 represents the lower bound, as it triggers exactly one inference step. Pipeline 4, in contrast, invokes the model three times, which increases latency and energy cost but provides improved answer grounding through diversified queries and snippet filtering. The filtering step is likely the decisive cost factor, since it receives the largest raw input of retrieved snippets. However, this step also reduces the final context length for the subsequent answer-generation call. Because both Pipeline 1 and Pipeline 4 ultimately perform answer generation on the provided context, but Pipeline 4 uses a *much smaller filtered context*, the additional filtering overhead can be partially compensated by lower token counts in the last step.

Given that the underlying model (`Qwen3-30B-A3B`) is a Mixture-of-Experts (MoE) architecture with 128 experts and 8 activated per token [6], its effective inference cost per token is already substantially reduced compared to dense alternatives. Recent work [16] shows that activating 8 experts instead of running a dense model can save 31–52% inference cost depending on the comparison baseline. Thus, while Pipeline 4 requires multiple calls, the underlying MoE structure amortizes per-call cost, and the filtering-induced context reduction helps balance the overall expense compared to Pipeline 1.

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 non-tied pairwise votes across the 20 prompts and shows a *marginal* but significant average advantage for **Pipeline 4** over **Pipeline 1**. As defined in Section 3.4, "win rate" reflects the fraction of comparisons in which a pipeline's answer was preferred under our annotator guidelines (Section 3.5).

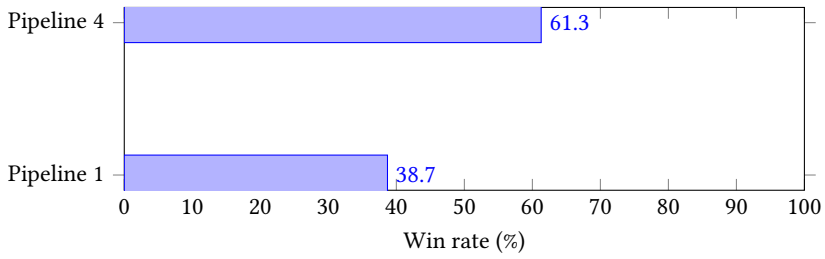

Fig. 3. Win rate comparison between Pipeline 1 and Pipeline 4.

Retrieval coverage limits. Not infrequently, Elasticsearch retrieved *no accurate, on-topic snippets* for a given prompt. In such cases, answers could not be fully grounded in the provided context, compressing the attainable performance of either pipeline and dampening any advantage from Pipeline 4's broader query coverage.

Hallucinations on sensitive topics. We observed instances where the model answered *regardless* of available context, i.e., hallucinated content, particularly for highly sensitive topics (e.g., COVID-19, climate change). This behavior violates the "answer only from snippets" instruction and directly harms the **Correctness** criterion, introducing 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.

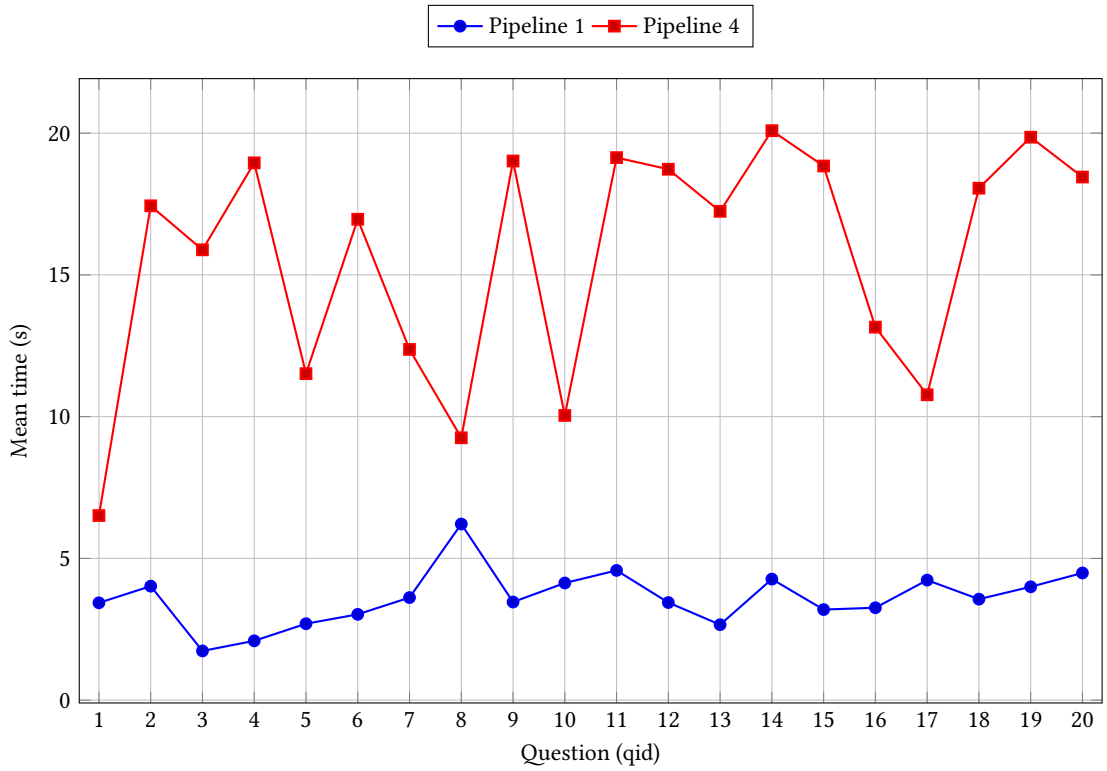

Fig. 4. Mean LLM response times per question (s) for the two pipelines.

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

xxx

Acknowledgments

To Maximilian, thanks for nothing.

References

[1] 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

[2] 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

[3] 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

[4] 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

[5] 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

[6] Tencent AI Lab. [n. d.]. Qwen3 30B A3B Model. https://huggingface.co/Qwen/Qwen3-30B-A3B

[7] 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

[8] 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/

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

[10] 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

[11] 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

[12] 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

[13] 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

[14] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, YaGuang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Vincent Zhao, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Pranesh Srinivasan, Laichee Man, Kathleen Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Soraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Diaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravi Rajakumar, Alena Butryna, Matthew Lamm, Viktoriya Kuzmina, Joe Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Aguera-Arcas, Claire Cui, Marian Croak, Ed Chi, and Quoc Le. 2022. LaMDA: Language Models for Dialog Applications. arXiv:2201.08239 [cs.CL] https://arxiv.org/abs/2201.08239

[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] 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

[17] 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