UNIVERSITY OF HELSINKI
FACULTY OF ARTS
DEPARTMENT OF DIGITAL HUMANITIES

Master's Thesis

Towards Optimizing the Retrieval Quality of a Retrieval-Augmented Chatbot for HPC User Support

Mitja Sainio

Master's Programme in Linguistic Diversity and Digital Humanities

Supervisors: Sebastian von Alfthan & Mathias Creutz 22.5.2025

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Tiedekunta – Fakultet – Faculty	Koulutusohjelma – Utbildningsprogram – Degree Programme
Faculty of Arts	Master's Programme in Linguistic Diversity and Digital Humanities

Opintosuunta – Studieinriktning – Study Track
Language Technology

Tekijä – Författare – Author
Mitja Sainio

Työn nimi – Arbetets titel – Title
Towards Optimizing the Retrieval Quality of a Retrieval-Augmented Chatbot for HPC User Support

Työn laji – Arbetets art – Level	Aika – Datum – Month and year	Sivumäärä– Sidoantal – Number of pages
Master's thesis	May 2025	48 pages and 3 appendices

Tiivistelmä – Referat – Abstract

High-performance computing (HPC), the use of supercomputers for solving complex computational problems, is behind an increasing number of significant advancements across diverse fields of science. Recent breakthroughs in machine learning that leverage deep neural networks have increased the range of available methods in fields that have traditionally benefited from HPC, like bioinformatics and computational chemistry, while also attracting new users from the fields of humanities and social sciences. Although a welcome development, the influx of users does also present challenges. Many new users find that the learning curve of using HPC systems is steep, which increases demand for expert support. This study investigates the use of retrieval-augmented generation (RAG) for complementing the support provided by HPC specialists with a chatbot that retrieves relevant information from a knowledge base of HPC documentation.

The study presents a comprehensive evaluation of a RAG system for HPC question answering. The objective of this evaluation is to determine how the system could be modified to improve the usefulness of retrieved documents. The evaluation is supported by curating a dataset of real user questions and expert answers, and identifying a set of appropriate metrics in previous work on text generation evaluation. Hyperparameters that control relevant parts of the system are varied in an effort to discern how they affect the usefulness of the documents retrieved for informing the generation process. Through comparing the generated responses of differently configured systems, it is found that the tested hyperparameters do not significantly improve retrieval quality, contrasting previous work.

The results highlight that the optimal configuration for a RAG system is dependent on the knowledge base from which information is retrieved, as well as the information needs that system is intended to satisfy. Closer analysis of the responses generated by different systems suggest that the advanced retrieval techniques typically used for improving the retrieval component are not as effective when applied to modestly-sized knowledge bases consisting of a few thousand documents. The study also provides insights into the limitations of text generation metrics that are commonly applied to RAG evaluation, as well as the potential benefits of leveraging cross-system documents for answering questions about specific HPC systems.

Avainsanat – Nyckelord – Keywords
HPC, language technology, NLP, RAG, retrieval-augmented generation, retrieval quality

Säilytyspaikka – Förvaringställe – Where deposited
E-thesis

Muita tietoja – Övriga uppgifter – Additional information

Contents

Chapter 1

Introduction

1.1 Motivation

High-performance computing (HPC) plays a key role in many scientific fields. Large-scale simulations and machine learning utilizing supercomputers have enabled significant advances in a variety of fields (Abas et al., 2024; Dongarra & Keyes, 2024; Kannagi et al., 2024; Min et al., 2023). Recently, advances in artificial intelligence have led to the increased adoption of computationally demanding methods across a variety of fields (Ettifouri et al., 2024; Su et al., 2021). As researchers' needs for computational resources grow, so does the appeal of HPC systems, and organizations that provide access to such systems can expect their user base to increase significantly. Since supercomputers are significant investments for these organizations, an influx of new users is a welcome phenomenon, as it increases the impact of these investments. However, this also creates challenges. For many researchers, HPC systems have steep learning curves, thus new users tend to have the greatest need for expert support.

Most information needs of new users are typically easier to meet than those of more advanced users, which makes supporting new users an attractive target for automatization. Recent advances in generative language modeling have prompted many providers of different services to utilize chatbots to complement human specialist support (Huang et al., 2024; Shahzad et al., 2024). Successfully directing a part of the user support workload to a chatbot allows organizations to increase their service level and frees up the time of human specialists for more complex support cases. While the large language models (LLMs) that power modern chatbots are able to engage with humans using fluent natural language and effectively process information that they are provided, they lack in-depth knowledge of many topics. Recently, this issue has been addressed using a technique called retrieval-augmented generation (RAG) (P. Lewis et al., 2020). RAG involves complementing the internal, learned knowledge of LLMs by retrieving documents from a knowledge base and using them as context for text generation.

Software libraries such as LlamaIndex[1] and Langchain[2] make building basic RAG systems easy. While the default configurations of these libraries typically offer a reasonable starting point, it is worthwhile to investigate whether tuning system hyperparameters could lead to further improvements. Recent work (Şakar & Emekci, 2025; X. Wang et al., 2024) has shown that optimizing RAG systems can have a significant effect on performance. While these works have largely sought to identify a universal set of best practices for RAG, optimization for a particular use case should not be overlooked. The information needs of the intended user as well as the composition of the RAG knowledge base ultimately determine how the latter should be used to augment the LLM.

RAG systems are considered to involve two distinct components: retrieval and generation (Yu et al., 2025). Of these two, optimizing the retrieval component is especially important. The only advantage that RAG offers over the use of a non-augmented LLM is access to additional information. Thus, while retrieval augmentation has the potential to be highly beneficial for an LLM, its effect is dependent on the retrieved information. While any issues in the retrieval component are propagated to the generated outputs, the reverse is not true, since modifications made to the generation component only affects how this additional information is used (Y. Gao et al., 2024). For these reasons, the focus of this study is on retrieval quality, which refers to the RAG system's capacity to retrieve information that is useful for the generation process.

Evaluating the performance of RAG systems is a challenging research problem due to the open-ended nature of the task (Afzal et al., 2024; Yu et al., 2025). However, since a reliable evaluation framework is necessary for optimizing a system, it is also an attractive one. Researchers and industry practitioners have developed a multitude of evaluation frameworks, and this line of research is still very active (Yu et al., 2025). Most of the metrics involved use LLMs to evaluate the system. This is typically done either by prompting the model to assign numeric scores to system outputs (e.g., TruLens[3]) or by using it to perform some natural language processing (NLP) task, which is then used to obtain a numeric assessment (e.g., Ragas) (Es et al., 2024). While LLMs have an impressive capacity for processing information, they are ultimately trained for next-token prediction and thus should not be relied on for numeric or binary outputs without scrutiny. Thus, standard text generation metrics, such as BARTScore (Yuan et al., 2021) and sentence embedding similarity (Reimers & Gurevych, 2019), merit consideration as alternatives.

This study investigates the optimization of a RAG system that powers an HPC user support chatbot. The chatbot is developed by CSC – IT Center for Science Ltd.[4], which is a Finnish company entrusted with special state assignment. Among other tasks, CSC offers HPC services

[1] `https://www.llamaindex.ai/`
[2] `https://www.langchain.com/`
[3] `https://www.trulens.org/`
[4] `https://csc.fi/en/`

to Finnish higher education and research. In addition to maintaining two systems exclusively for Finnish users, CSC also hosts the pan-European supercomputer LUMI[5]. Consequently, the company serves a large number of users and therefore stands to benefit greatly from complementing expert support with a retrieval-augmented chatbot. The outcomes of this study mainly contribute to the development of this chatbot. However, they also provide insights into applying RAG to the domain of HPC and conducting reference-based end-to-end RAG evaluation. To support the transparency and reproducibility of the study, the code and data used in the experiments have been made available at `https://github.com/msainio/hpc-rag`.

1.2 Research Questions

The following research questions are considered in this study:

- **RQ1: Which RAG hyperparameters lead to the greatest improvement in retrieval quality for HPC question answering?**
- **RQ2: To what extent are the selected evaluation metrics correlated with one another, and with the length ratio of generated responses to ground truth answers?**
- **RQ3: How does retrieving cross-system document passages affect retrieval quality?**

RQ1 involves determining which hyperparameters of the RAG system have the greatest positive impact on retrieval quality when tuned. In practice, this is approached by building a number of differently configured systems, evaluating each of them and using statistical analysis to determine whether their retrieval quality differs in a significant manner. The evaluation of these systems involves creating an HPC question answering dataset and determining a set of metrics for quantifying retrieval quality. All data used for evaluation are anonymized in order to protect the personal information of data subjects.

RQ2 entails measuring the strength of the pairwise correlation between evaluation metrics and length ratio of generated responses to reference answers. The relationships indicated by correlation analysis help interpret the evaluation results of RQ1, reveal any complementarity or redundancy in the metrics, as well as their potential dependence on factors unrelated to retrieval quality, as exemplified by response length ratio. RQ3 concerns assessing whether retrieving the documentation of one HPC system is beneficial for answering questions about another. This question is studied by dividing both the evaluation questions and data sources into CSC and LUMI groups, and measuring the correlation between the evaluation metrics and the proportion of cross-group items in a set of retrieved documents.

[5] `https://www.lumi-supercomputer.eu/`

Chapter 2

Background

2.1 Applications of RAG in HPC Question Answering

Few studies have been conducted on question answering (QA) in the HPC domain, let alone the use of RAG for this purpose. The development of the HPC-GPT model by Ding et al. (2023) appears to be one of the earliest works on HPC QA. They fine-tuned Llama models (Touvron, Lavril, et al., 2023; Touvron, Martin, et al., 2023) on HPC-related data obtained from GitHub, scientific papers and websites. While HPC-GPT exhibited higher performance than the base LLMs it was compared to, Ding et al. (2023) noted that maintaining the model up to date is a challenge, suggesting RAG as a potential solution.

According to available knowledge, Biggs and Dalton (2024) are the first to apply RAG to HPC QA. They integrated a RAG chatbot into the graphical user interface of their HPC systems. The chatbot retrieved document passages from a knowledge base built from educational materials and internal documentation, feeding them as context to Vicuna (Chiang et al., 2023), an instruction fine-tuned Llama 2 model (Touvron, Martin, et al., 2023). Unfortunately, since Biggs and Dalton (2024) did not report having conducted an evaluation of the model and did not present the user feedback they gathered, this did not contribute to a better understanding of the usefulness of RAG for HPC QA.

This knowledge gap was addressed by Yin et al. (2024), who compared fine-tuning and RAG-based approaches for implementing an HPC user support chatbot. Using HPC documentation and user support tickets as data sources, they built an instruction dataset and a RAG knowledge base. Moreover, they curated an evaluation set out of a subset of their available support tickets. Conducting a multifaceted evaluation of their systems, Yin et al. (2024) found RAG to significantly improve the quality of system generations in the QA task. They concluded that their fine-tuned retrieval-augmented LLM is an effective solution to answering questions on HPC documentation.

2.2 Optimization of RAG Systems

Comprehensive studies have been carried out on tuning the hyperparameters of RAG systems (Şakar & Emekci, 2025; X. Wang et al., 2024). These typically aim to provide a generally applicable set of best practices. However, Saad-Falcon et al. (2024) note that the domain and size of the available knowledge base, as well as tolerance for cost and latency, are necessary considerations when designing the optimal configuration for a RAG system. This has prompted many evaluation frameworks to incorporate the generation of synthetic datasets based on documents in the knowledge base (de Lima et al., 2024; Es et al., 2024; Saad-Falcon et al., 2024; Zhu et al., 2025). There is also a fair-sized body of work on optimizing RAG systems for domain-specific use cases (Afzal et al., 2024; Zhao et al., 2024). These lines of inquiry highlight the importance of evaluating RAG systems in a way that reflects their intended purpose as closely as possible.

2.3 Evaluation of Retrieval-Augmented Generation

While RAG systems are generally considered to involve two distinct components, retrieval and generation, most existing RAG evaluation frameworks assess the system as a whole (Yu et al., 2025). Evaluation is typically focused on some variation of the three qualities of faithfulness, answer relevance and context relevance. These are used as the basis of many frameworks (Yu et al., 2025), such as Ragas (Es et al., 2024), ARES (Saad-Falcon et al., 2024) and TruLens (TruLens, 2025). While these systems may share their evaluation targets, they measure them in diverse ways. Ragas uses an LLM to perform NLP tasks, whose outputs are used for scoring. For example, faithfulness is evaluated by extracting a set of statements from a text, and assigning binary labels to them based on whether they are supported by retrieved context information. ARES trains a classifier on LLM annotations for predicting these qualities, while TruLens simplifies this process by simply prompting an LLM to rate the generated text on a numeric scale.

All these frameworks are notable for being reference-free. Unlike traditional text generation evaluation metrics such as BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004), they do not require ground truth outputs for comparison. This is likely to be a contributor to their popularity. While the evaluation of a RAG application should reflect its intended real-word usage (Yu et al., 2025), such target-specific datasets are typically not readily available (Es et al., 2024). While reference-free evaluation frameworks are straightforward to apply, their commonplace reliance on LLMs for numeric evaluation is not entirely unproblematic. While techniques such as chain-of-though prompting (Wei et al., 2022) improve the interpretability of LLM-as-a-judge evaluations (Zheng et al., 2023), they are not standardized or entirely consistent.

Because of the limitations of reference-free frameworks, many RAG benchmarks, such as

CRUD-RAG (Lyu et al., 2025) and RECALL (Y. Liu et al., 2023), employ reference-based evaluation using standard text generation metrics (Yu et al., 2025). This type of evaluation is typically conducted end to end, which means that the quality of both retrieval and generation are evaluated based on generated text output. For example, in order to optimize the retrieval component, the hyperparameters affecting retrieval are modified, and generation quality used as a proxy for retrieval quality. This is the approach taken by Lyu et al. (2025), among others.

Despite the popularity of end-to-end evaluation, Salemi and Zamani (2024) argue that it is not a transparent measure of retrieval quality, since it does not distinguish the individual contributions of retrieved documents. Their proposed framework, eRAG, addresses this by generating multiple system outputs for each query, using the retrieved documents one-by-one. The outputs are then evaluated separately using reference answers, and their scores aggregated using standard information retrieval metrics, such as precision and recall. While eRAG seeks to improve the relevance assessment of individual documents, it may not be ideal when integrating knowledge from multiple sources is necessary. For example, certain tasks in the CRUD-RAG benchmark (Lyu et al., 2025) require using several documents for answering questions. If a set of superficially less relevant documents form an informative whole for an LLM, evaluating their usefulness separately is not ideal. Considering this, end-to-end evaluation is not necessarily less reliable than eRAG for evaluating retrieval quality.

Common metrics used for end-to-end evaluation are traditional text generation metrics such as BLEU, ROUGE and BERTScore (T. Zhang et al., 2020) (Yu et al., 2025). These are used in Afzal et al. (2024) and Lyu et al. (2025), for example. Although they are more consistent evaluators than LLMs-as-judges, their usefulness for evaluation of free-form QA, the core task of RAG (Y. Gao et al., 2024), is limited due to the fact that they operate on the level of tokens (Becker et al., 2024; A. Chen et al., 2019). While more recent text generation metrics, such as BARTScore (Yuan et al., 2021), are not widely used for RAG evaluation, there are exceptions, such as the adaptation of the QuestEval summarization metric (Scialom et al., 2021) into the RAGQuestEval of Lyu et al. (2025). Since these metrics offer a more cost-efficient alternative to resource-intensive LLMs, their use for RAG evaluation is an attractive area of study (Yu et al., 2025).

2.4 Evaluation of Long-Form Question Answering

Long-form question answering (LFQA) (Fan et al., 2019; Krishna et al., 2021) is the task of retrieving documents that are relevant to a question and using them to produce a paragraph-length answer. It is a common task for RAG alongside the less open-ended task of extractive QA (Y. Gao et al., 2024), where the answer consists of a short phrase or a single entity. RAG evaluation datasets like WikiEval (Es et al., 2024) and RAGBench (Friel et al., 2025) involve precisely this task. Considering that LFQA is a key benchmark for RAG, previous work on LFQA can provide meaningful evaluation metrics.

Like RAG evaluation, automatic evaluation of LFQA remains an open research problem (Xu et al., 2023). BLEU and ROUGE (Fan et al., 2019; Kočiský et al., 2018) still used as metrics, although it has been shown that they have a weak correlation with human preferences for evaluating QA (A. Chen et al., 2019). Neither metric is intended for this application: BLEU is meant to be used for evaluating machine translation and ROUGE for summarization. BERTScore, which uses an encoder-only model to compare the individual token embeddings of two texts, has a higher correlation with human preferences , but is still limited by its token-level approach (Becker et al., 2024).

Some more flexible model-based evaluation methods have been proposed for other text generation tasks, such as summarization. While there is little work on their use for LFQA evaluation, the properties they measure are, in principle, suitable (Xu et al., 2023). Recent metrics for evaluating the factuality of summaries typically leverage established NLP tasks, most prevalently natural language inference (NLI) (Dagan et al., 2013) and QA (Koh et al., 2022). NLI-based metrics, such as SummaC (Laban et al., 2022), use NLI classifier models for reference-free evaluation by predicting the probability of a source text entailing a summary.

The application of NLI to summarization evaluation is based on the rationale that all claims in a summary should be entailed by the source text. This is similar to the evaluation of faithfulness carried out by the previously mentioned reference-free RAG evaluation frameworks. MENLI (Y. Chen & Eger, 2023) extends the use of NLI to reference-based general evaluation of text generation by comparing candidate and reference summaries. This approach translates well to reference-based QA evaluation, and indicates the potential of applying existing NLP text generation metrics to evaluation of RAG.

Chapter 3

Methodology

3.1 Data

3.1.1 Knowledge Base

Considering that the only difference between RAG and non-augmented generation is the utilization of retrieved document passages as additional context, it is fair to say that a RAG system is defined by its knowledge base. The knowledge base used in this study consists of CSC and LUMI materials that are openly available on GitHub. The following sources are used for the knowledge base: user guides and tutorials for CSC services (`csc-user-guide`)[1], materials for the CSC Computing Environment course (`csc-env-eff`)[2], the LUMI supercomputer user guide (`LUMI-userguide`)[3] and the documentation of the LUMI software stack (`LUMI-EasyBuild-docs`)[4].

Since the system architecture and usage policies of LUMI differ considerably from those of the two national CSC supercomputers Puhti and Mahti, the four data sources are divided into two groups: the CSC group and the LUMI group. The purpose of this division is to determine whether a document passage retrieved for a question is same-system or cross-system. For example, while more general HPC questions could be answerable using documents from either group, questions about the specifics of the Puhti system architecture (e.g., the number of GPU nodes) is not likely to be answerable using the cross-system LUMI documentation. In this study, the CSC group includes `csc-user-guide` and `csc-env-eff`, while the LUMI group includes `LUMI-userguide` and `LUMI-EasyBuild-docs`.

The knowledge base is constructed from repository archives that are automatically generated by GitHub. The archives are downloaded and extracted locally, after which irrelevant files are

[1] https://github.com/CSCfi/csc-user-guide
[2] https://github.com/csc-training/csc-env-eff/
[3] https://github.com/lumi-supercomputer/lumi-userguide
[4] https://github.com/Lumi-supercomputer/LUMI-EasyBuild-docs

removed. The latter includes images as well as scripts used in hosting the documentation online. All data sources are obtained in this straightforward manner, except for `LUMI-EasyBuild-docs`, which must be generated with a script that requires data from the `LUMI-EasyBuild-containers`[5], `LUMI-EasyBuild-contrib`[6] and `LUMI-SoftwareStack`[7] repositories. All of the documents that are used in the knowledge base are Markdown files. Table 3.1 shows the number of files from each data source.

Data source	Files
`LUMI-EasyBuild-docs`	4,373
`csc-user-guide`	612
`lumi-userguide`	71
`csc-env-eff`	69
Total	5,125

Table 3.1: Number of files from each data source of the knowledge base.

3.1.2 Evaluation Dataset

In order to assess system retrieval quality, a dataset of QA pairs concerning CSC services and LUMI was created. Two kinds of data sources were considered for the evaluation set. The first data source comprises the weekly remote user support meetings hosted by CSC and LUMI staff. Before these meetings, users may write any questions they may have in a collaborative document. Staff members can then familiarize themselves with user questions in advance and record their answer below the question once it has been discussed at the meeting. Once a document reaches a certain number of questions, it is archived and a new one is opened.

The archived user questions and expert answers are, in principle, a good data source for an evaluation dataset, since they represent actual user needs and contain real expert answers. However, due to the fact that answers are provided to the users during the meetings, they are not always recorded. Still others are simply links to relevant documentation, or, if the question is sufficiently advanced, contain information that is so specialized that it cannot be found in the knowledge base.

The second source that was considered is the various virtual trainings organized by CSC and LUMI staff. Similarly to the weekly support meetings, the trainings make use of collaborative documents. Unlike in the weekly support sessions, there is no allotted time for answering participant questions, so any answers are provided only in writing. Since the written answers must be useful by themselves, rather than complementing a live discussion, a greater proportion of the answers are useful compared to those in the support meeting documents. Additionally, since the

[5] `https://github.com/Lumi-supercomputer/LUMI-EasyBuild-containers`
[6] `https://github.com/Lumi-supercomputer/LUMI-EasyBuild-contrib`
[7] `https://github.com/Lumi-supercomputer/LUMI-SoftwareStack`

trainings are aimed at new users, the questions presented there are typically less complex than those presented in the user support meetings, and thus may be more readily answered using the documentation that comprises the knowledge base. Due to these properties, the collaborative documents from the trainings were chosen as the data source for the evaluation dataset.

The dataset was constructed from the collaborative documents of 13 trainings organized in 2023 and 2024. The trainings in question are the CSC Computing Environment courses and LUMI 1- and 2-day courses, which correspond to the CSC and LUMI groups of the knowledge base. The CSC documents are published under a CSC expert's account on HackMD[8], while the QA pairs from the LUMI documents are stored in a GitHub repository[9] along with notes and links to training materials. All of the documents are in the Markdown format.

While many of the CSC documents could be downloaded as Markdown files through HackMD, for some this option was not present. The latter were then scraped as HTML documents, from which the Markdown content could then be extracted. Since both CSC and LUMI documents contained additional data, such as training schedules, notes and links, regular expressions were used to extract the QA pairs from the documents. The expressions also removed all Markdown hyperlinks, leaving only the anchor text. This procedure produced a total of 470 QA pairs.

To ensure that each QA pair is suitable for evaluating the system, the collected set of pairs was curated manually. Many of the pairs contained references to the trainings themselves. These included, for example, practical instruction, as well as titles and contents of lectures and materials. Some questions and answers also contained names of participants and instructors. Although the contents of the collaborative documents are openly available online, the experiments involve using models that are not self-hosted or open-source, but rather accessed through the OpenAI API[10]. Since data submitted through the API may be processed by the model provider for training purposes, all names were removed from the data in order to protect the personal information of data subjects. When references to trainings could not be removed from a QA pair without rendering it meaningless, the pair was omitted from the dataset (Table 3.2). In other cases, the pair could simply be edited slightly (Tables 3.3 and 3.4).

Even if the information content of the pair did not require modification, some manual curation was necessary in all QA pairs in order to remove Markdown metacharacters and other artifacts. Questions and answers were typically prefixed with a dash so that they would be displayed as bulleted lists or they contained asterisks for boldfacing the text. While the regular expressions used for extracting the pairs removed leading metacharacters from questions, trailing asterisks were still present in most of them. The metacharacters in the answers were not removed by the regular expressions.

For many questions, multiple answers were provided. All answers of sufficient quality were

[8] https://hackmd.io/@CSCBioMaria
[9] https://github.com/Lumi-supercomputer/LUMI-training-materials
[10] https://openai.com/api/

Question	Answer	Reason for omission
Should we be able to access **the slides in e-elena**? I can only see the first slide.**	- A: Try navigating with the arrow keys :)	Reference to training material
I need to leave the session unfortunately, can you post the prep for next day instruction somewhere? Thank you it would be great if this is possible.**	- A: Thank you for joining today! See you tomorrow at 9:00, same Zoom URL :) No specific preps needed!	Practical instruction
I didn't understand **the part about the line endings**. Is this something we need to worry about if we're using powershell to access puhti or?**	- A: It something that is good to be aware of. Sometimes when you create files with Windows there may be hidden characters added at the end of lines that will cause things to fail on Linux. Converting with 'dos2unix' will solve these issues. So if you get strange errors and are not sure why, 'dos2unix' is good to try! - Q: So this is something we need to worry about if we created a batch job file (or any other one) within windows that we then want to import and run on Linux, yes? - A: Exactly.	Implicit reference to lecture

Table 3.2: Examples of omitted QA pairs.

Question	Edited question	Reason for editing
how can we check **for all these points mentioned?** e.g. if cores are waiting, How do we know that, and then how do we address that? **and the rest of the points as well, how do we test or check for them?** If the seff doesn't see that all the memory is used, how would I know that I'm running out of memory and ralize that I need to request more?**	how can we check e.g. if cores are waiting, How do we know that, and then how do we address that? If the seff doesn't see that all the memory is used, how would I know that I'm running out of memory and ralize that I need to request more?	Implicit reference to lecture
Are containers and images the same thing? What about dockers and tykky? A place where all the small files are treated as one large one if I understand correcty. What was the container registry/container wrapper **that was mentined in the lecture yesterday. There was a lot of new terminology for me and I didn't understand what was downloaded from the quay.io website.****.	Are containers and images the same thing? What about dockers and tykky? A place where all the small files are treated as one large one if I understand correcty. What is a container registry/container wrapper?	Reference to lecture

Table 3.3: Examples of edited questions.

retained. Their lines were merged and they were separated with the line break string "\n". The second text in Table 3.4 provides an example of this. The answer "How large is huge?" was excluded from the curated answer, since the participant did not respond to the inquiry, and as such it did not help answer the question.

In some pairs, the user had asked a follow-up question after the answer, which were typically also answered. If a follow-up question did not require the original question as context for understanding it, a new QA pair was formed from the follow-up question and its answers. When this was not the case, these were either merged with the original question and answer(s) or removed completely. After manual curation, the number of QA pairs was 300. Table 3.5 displays the number of QA pairs from each each training before and after curation. The curation process increased the proportion of LUMI QA pairs by five percentage points. Nonetheless, the majority of the QA pairs were from CSC trainings.

Question	Answer	Edited answer	Reason for editing
What is the price for billing units in commersial projects?	**- A: More about pricing of commercial projects: https://research.csc.fi/purchasing** The base package costs 1190 EUR (VAT 0 %). It includes: 20 000 BUs, 4 user accounts. Note that LUMI has a different (more affordable) pricing**: https://www.csc.fi/solutions-for-business**	The base package costs 1190 EUR (VAT 0 %). It includes: 20 000 BUs, 4 user accounts. Note that LUMI has a different (more affordable) pricing	Answer contains URLs
Which disk area shall I use if I'm reading and writing a lot of huge files (like several 5-10 GB fil)?Would $LOCAL_SCRATCH bring a performance boost?	**- A:** How large is huge? **-** scratch is the correct one. **-** You can try local_scratch. It may be faster, depending on what is the bottleneck of your work.	scratch is the correct one.`\n`You can try local_scratch. It may be faster, depending on what is the bottleneck of your work.	Combining multiple answers

Table 3.4: Examples of edited answers.

The lengths of questions and answers are similar within the CSC and LUMI groups (Figure 3.1). The average length of both questions and answers is higher in the LUMI group, but the difference is only that of a few words. The distributions of the lengths have approximately the same form, although long answers (those in excess of 160 words) are more frequent in the LUMI group. In conclusion, there is not much superficial difference between the QA pairs of the two groups. The texts were tokenized for length measurement using the spaCy Python library (Honnibal et al., 2020).

3.2 RAG Hyperparameters

The program code that is used for augmenting an LLM with a knowledge base is referred to as a RAG pipeline. The pipeline used in this study is built using the LlamaIndex[11] Python library.

[11] `https://www.llamaindex.ai/`

Source	Raw	Curated
CSC		
`csc-enveff-20230412`	33	21
`csc-enveff-20230919`	34	17
`csc-enveff-20231128`	30	15
`csc-enveff-20240207`	51	29
`csc-enveff-20240424`	53	33
`csc-enveff-20240515`	34	24
`csc-enveff-20241003`	32	16
`csc-enveff-20241107`	22	12
Total	289	167
LUMI		
`lumi-1day-20230509`	57	37
`lumi-1day-20230921`	32	24
`lumi-1day-20240208`	43	35
`lumi-2day-20240502`	5	1
`lumi-2day-20241210`	44	36
Total	181	133

Table 3.5: Number of QA pairs before and after curation.

LlamaIndex provides the basic tools for transforming the knowledge base into a searchable index, retrieving information from this index and feeding the retrieved information to the system's generator LLM as context. There are also implementations for various advanced techniques that may be used to adjust these processes.

The RAG pipeline presents many opportunities for modification. In this study, the configurable parts of the pipeline are referred to as hyperparameters. The RAG hyperparameters and their values used in the experiments were chosen from the indexing and retrieval hyperaparameters identified by X. Wang et al. (2024). A key consideration in choosing the possible values was not increasing the number of LLM calls involved in the pipeline. LLM calls are computationally expensive, so they would likely increase system latency (X. Wang et al., 2024). In practice, this policy excludes query transformations such as HyDE (L. Gao et al., 2023).

3.2.1 Indexing

This section details the indexing hyperparameters and explains how the index is built. Figure 3.2 (Askaryan, 2023) visualizes the RAG indexing process.

Chunk Size and Chunk Overlap

At the start of the indexing phase of the RAG pipeline, the documents in the knowledge base are first loaded into system memory and split into passages of a certain length in a process called chunking. While state-of-the-art LLMs, such as the Llama 3 range of models (Llama Team,

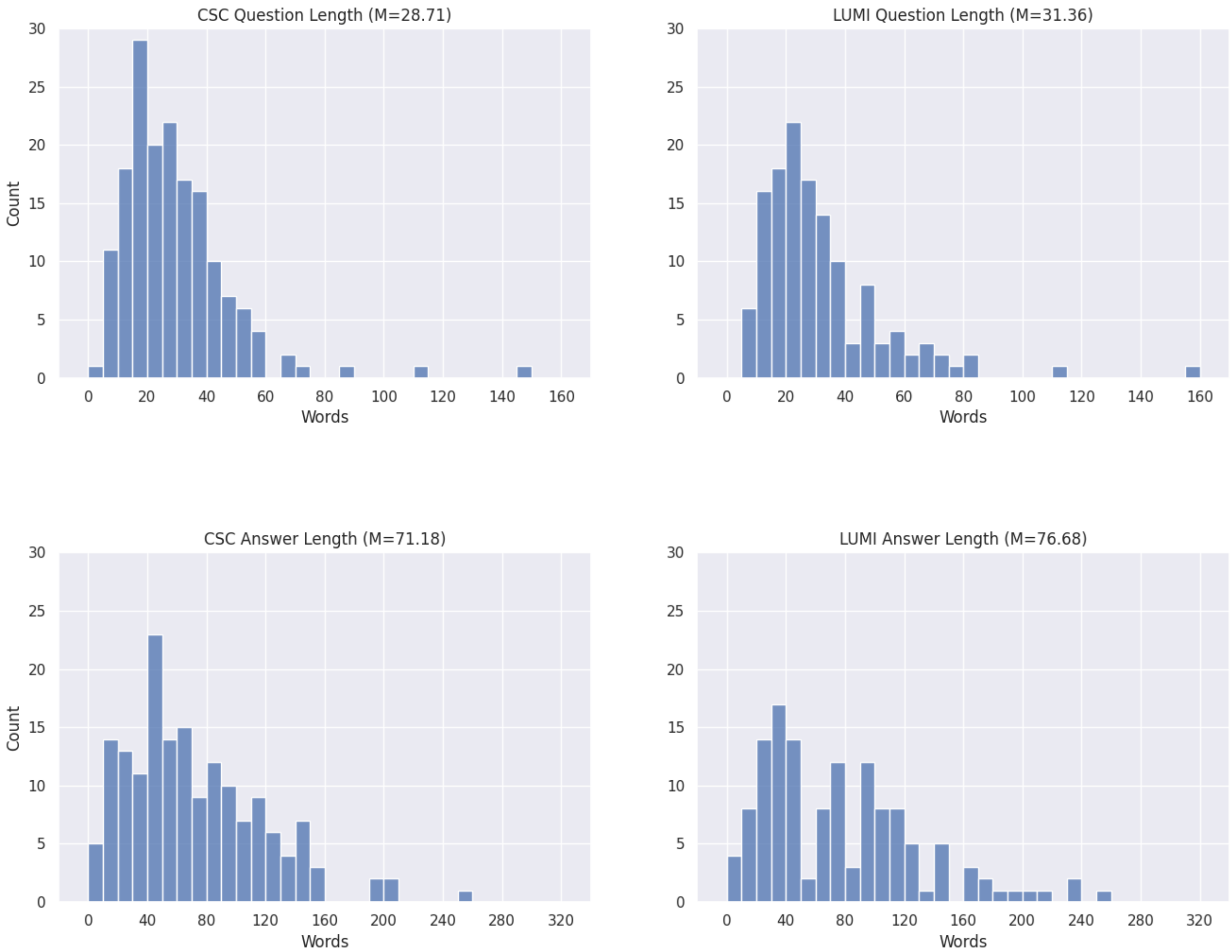

Figure 3.1: Histograms of question and answer lengths of CSC and LUMI QA pairs.

2024), have context windows of over a hundred thousand tokens, the amount of content they are able to effectively utilize is limited (N. F. Liu et al., 2024). A RAG system typically feeds the LLM multiple items from the index, in which case complete documents might be too long, leading to loss of information.

LlamaIndex uses two hyperparameters to control chunking: chunk size and chunk overlap. Chunk size determines the amount of tokens in each document chunk, while chunk overlap determines how many overlapping tokens two consecutive chunks may have. Documents are typically tokenized for chunking using the generator LLM's tokenizer. Several chunk size and overlap values are used in the experiments.

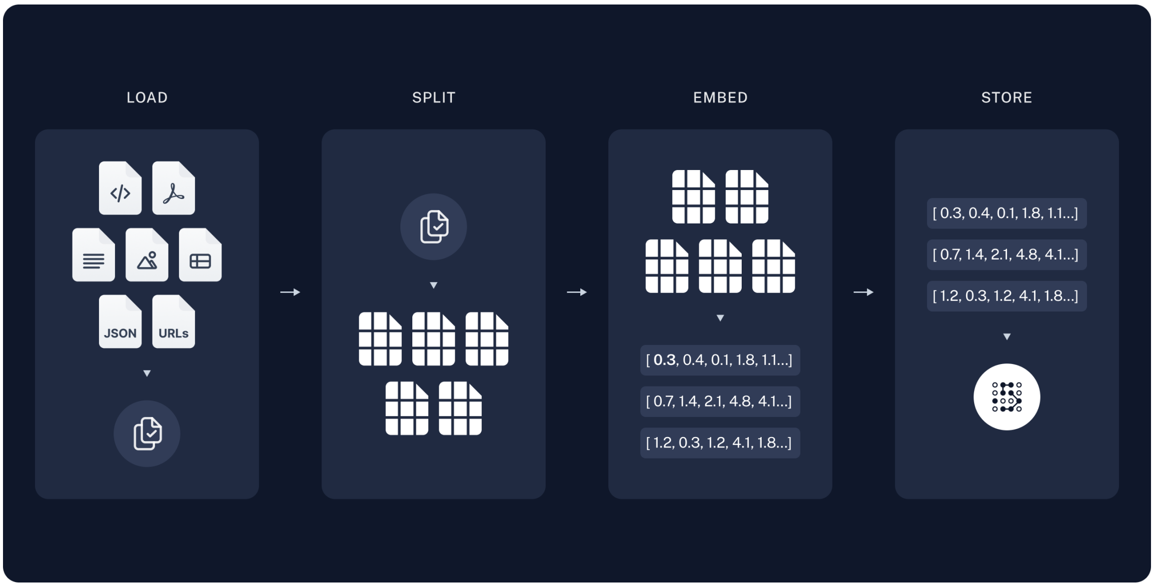

Figure 3.2: RAG indexing workflow. From [RAG indexing diagram] by B. Askaryan, 2023, GitHub (`https://github.com/langchain-ai/langchain/blob/master/docs/static/img/rag_indexing.png`). MIT License

Embedding Model

After the documents have been split into chunks, the chunks are uploaded into an index. A popular choice of index is the vector store index, where each document passage is stored along with a dense embedding computed using an encoder language model. When the index is queried, a new embedding is computed for the query, which is then compared against the embeddings in the index using a similarity function, such as cosine similarity or dot product. The chunks are then ranked according to their similarity scores.

While LlamaIndex implements several object classes for indices, such as the vector store index, these are typically simple software wrappers. The underlying data structures, for instance, vector stores, are typically implemented using some third-party software. The vector store used in this study is the Qdrant vector store[12]. It is run as a Docker container (Merkel, 2014) and can be accessed using, for example, Qdrant's own Python client, as is done in this study.

The vector store is responsible for storing the embeddings, the associated document chunks, and any relevant metadata such as the path of the original document file. Additionally, it is the vector store that is tasked with retrieving the most similar items when provided with a query embedding. Thus, the main task of LlamaIndex in interacting with the vector store is embedding the document passages and queries using a third-party embedding model. LlamaIndex also defines the similarity function to be used for the index. In the experiments, the default function is used, which is cosine similarity.

[12] https://qdrant.tech/

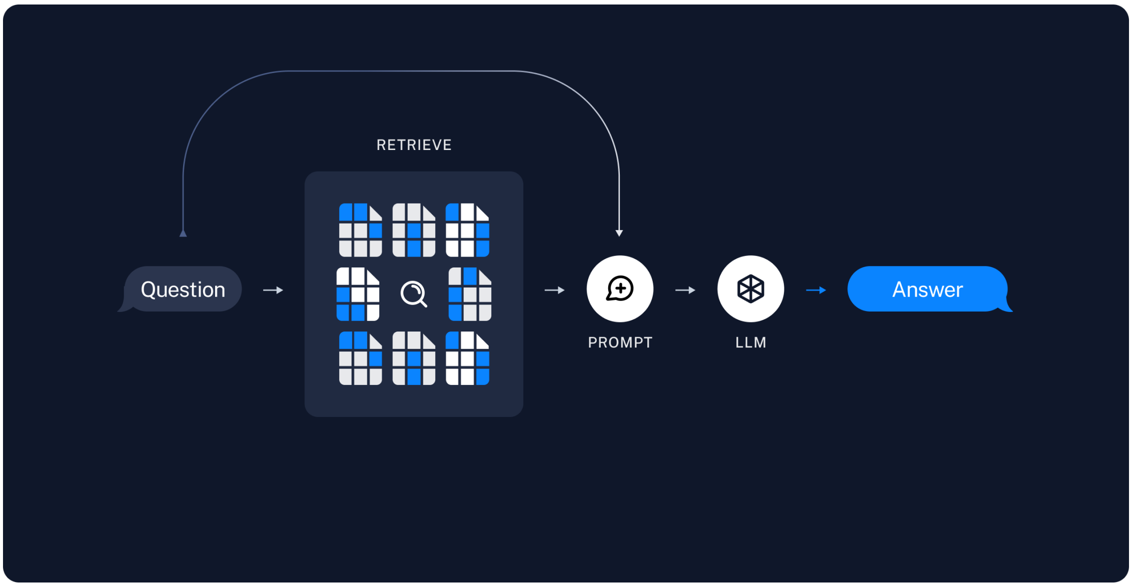

Figure 3.3: RAG retrieval and generation workflow. From [RAG retrieval and generation diagram] by B. Askaryan, 2023, GitHub (`https://github.com/langchain-ai/langchain/blob/master/docs/static/img/rag_retrieval_generation.png`). MIT License

Two text embedding models are used in the experiments: the OpenAI `text-embedding-3-small` and `text-embedding-3-large` (OpenAI, 2024b). The difference between these models is the dimensionality of the embeddings they produce. The small model can produce embeddings of up to 1536 dimensions, while the large model produces those of up to 3072. Embeddings of higher dimensionalities allow for fine-grained representations of the encoded text, while low-dimensional embeddings may compress the information content of long inputs. In the experiments, the maximum dimensionality was used for each model.

3.2.2 Retrieval

This section presents the retrieval hyperparameters as well as how the RAG system works at runtime. Figure 3.2 (Askaryan, 2023) visualizes the retrieval process.

Search Type

While the vector index is a common choice for RAG, other indices may also be used either by themselves or for complementing a vector index. The latter approach is termed hybrid search. Typically, hybrid search involves using a vector index in tandem with a keyword index built for a lexical ranking algorithm (Y. Gao et al., 2024). One such algorithm is BM25 (Robertson et al., 1995), which ranks documents based on how frequently each word in the query appears in them. Both indices retrieve a certain number of results, and their respective rankings are aggregated using a fusion algorithm.

The search types used in the experiments are pure vector search, pure BM25 search as well as hybrid search combining the two. In the experiments, different numbers of top-ranking results are retained from the two indices for aggregation. The LlamaIndex BM25 retriever is used for building the keyword index. It utilizes the BM25S Python library (Lù, 2024). The fusion algorithm used for hybrid search is reciprocal rank fusion (RRF) (Cormack et al., 2009), which combines individual index scores using a simple formula.

Re-Ranking

Vector indices rely on bi-encoder models, which compute text embeddings for single texts. This is in contrast to cross-encoder models, which compute embeddings for pairs of texts. Bi-encoders are ideal for searching through large text collections, since embeddings may be pre-computed for the texts, leaving only the query to be encoded at runtime. However, retrieval using cross-encoders is often more accurate. A widely-used compromise is the initial ranking of the entire collection using bi-encoder embeddings, and then re-ranking a certain number of the highest-ranked texts using the more expensive cross-encoder embeddings. Unlike the methods corresponding to the other hyperparameters, the use of re-ranking in the RAG pipeline is optional.

In the experiments, different numbers of top-ranking chunks are re-ranked using a 6-layer MiniLM model[13] (W. Wang et al., 2020) fine-tuned on the MS MARCO passage ranking dataset (Bajaj et al., 2018). This model was chosen because of its balance between performance and efficiency among the cross-encoder models pretrained by Sentence Transformers (Sentence Transformers, 2025a). The model's maximum input length is only 512 tokens, which means that many chunks may be truncated by the re-ranker's tokenizer. However, this is not an issue, since representing a long text document using just its beginning has been found to be a competitive approach (X. Zhang et al., 2021). As most questions in the evaluation dataset are quite short (Figure 3.1), most of the available tokens may be used for the document chunk.

Number of Context Passages

P. Lewis et al. (2020) use the variable *k* to refer to the number of document passages provided to the generator model of their RAG system. Most work on RAG thus refers to this hyperparameter as some variation of "top-*k*". In LlamaIndex, this variable refers to the number of chunks returned by any document index, including hybrid indices, while the variable *n* is used for the number of chunks returned by re-ranking. In this study, the number of context passages is

[13] `https://huggingface.co/cross-encoder/ms-marco-MiniLM-L6-v2`

Hyperparameter	Values
Augmentation	Yes, No
Chunk size	128, 256, 512, 1024, 2048
Chunk overlap	0, 20, 40, 80, 160, 320
Embedding model	Small, Large
Search type	Semantic; BM25; Hybrid, $k \in \{2,4,8,16\}$
Re-ranking	No; Yes, $k \in \{2,4,8,16\}$
No. of context passages	2, 4, 8, 16

Table 3.6: Hyperparameter values used in experiments. Default values used in the baseline system are underlined. The k in hybrid search and re-ranking refers to the number of chunks passed to the fusion algorithm and re-ranker model, respectively.

defined as the number of chunks that the generator LLM is provided. The number of context passages is varied in the experiments.

3.3 Experimental Setup

23 RAG configurations are created by varying the presented hyperparameters. Additionally, a non-augmented system consisting of just a generator LLM is introduced for comparison, making the total number of systems 24. The RAG systems differ from each other by only one hyperparameter, while default values are used for all other hyperparameters. Value ranges used for numeric hyperparameters, such as chunk size, are based on those used in Lyu et al. (2025) and X. Wang et al. (2024). Table 3.6 displays the full set of hyperparameters and their values.

The questions from the evaluation set are fed to the systems, whose outputs may then be evaluated against the corresponding ground truth answers in order to identify differences between the systems. The LLM used in each of the systems is GPT-4o mini (OpenAI, 2024a). The LLM's generation temperature is set to 0, which has the effect of assigning a greater weight to the most probable tokens. This decreases variation in the outputs and thus allows for a more reliable assessment of the effect of each hyperparameter.

3.4 Evaluation

An end-to-end approach is applied to evaluation of retrieval quality. This entails using the lexical, semantic and logical similarities between the generated responses and the ground truth answers as a proxy for the relevance of retrieved document chunks. This is based on the assumption that the more relevant the retrieved chunks are, the more the generated responses resemble the ground truth answers. This section details the metrics used for evaluating the responses. While each metric quantifies the similarity of two texts, their specific implementations vary. Using a diverse set of metrics allows for a robust assessment of retrieval quality.

In addition to the evaluation metrics, two additional properties of the system outputs are measured. Cross-system proportion measures the proportion of retrieved passages from data sources that do not belong to the same system group as the question. The other additional measurement is length ratio, which is computed by dividing the number of words in a generated response with that of the ground truth answer. The spaCy library (Honnibal et al., 2020) is used for word tokenization. The two measurements are not used to evaluate retrieval quality, but may be used to better understand the evaluation results through correlation analysis.

3.4.1 Lexical Metrics

BLEU (Papineni et al., 2002), ROUGE (Lin, 2004) and METEOR (Lavie & Agarwal, 2007) are lexical metrics. This means that they measure surface-level similarities between text. While their reliability for evaluating LFQA has been challenged (Gehrmann et al., 2023; Xu et al., 2023), they are still used as metrics in contemporary text generation and RAG benchmarks, such as CRUD-RAG (Lyu et al., 2025), and as such are useful for comparing experimental results to previous work. BLEU is computed using the sacreBLEU library (Post, 2018), ROUGE using the summary-level ROUGE-L implementation by Google Research (Google Research, 2024) and METEOR using the NLTK Python library (Bird et al., 2009).

3.4.2 Model-Based Metrics

BERTScore

BERTScore (T. Zhang et al., 2020) is a popular text generation evaluation metric that relies on an encoder model. It involves computing word embeddings for the tokens in two texts and matching each embedding to the most similar one in the other text. The final text-level similarity score is given by using the embeddings to compute precision, recall and an F-measure. The F-measure was used for evaluation, as it works for a variety of evaluation contexts T. Zhang et al. (2020). The BERTScore used for evaluation is the F-measure obtained using their implementation. The backbone encoder model is a large DeBERTa V1 model[14] (He et al., 2021) fine-tuned on the MultiNLI dataset (Williams et al., 2018). This is one of the two models recommended in the GitHub repository for the authors' implementation of BERTScore[15].

[14] `https://huggingface.co/microsoft/deberta-large-mnli`
[15] `https://github.com/Tiiiger/bert_score`

BARTScore

The BARTScore metric (Yuan et al., 2021) uses a sequence-to-sequence model to compute the log probability of generating a target text conditioned on a source text. Yuan et al. (2021) define four methods for using BARTScore. Three of these utilize a reference text and are therefore of interest for end-to-end evaluation: generation from reference text to hypothesis (precision), generation from hypothesis to reference (recall) and their arithmetic average (F-score).

In this study, the system-generated responses are used as hypotheses and the ground truth answers as references, while the method chosen is the F-score. The latter is chosen because Yuan et al. (2021) describe the F-score as a measure of the semantic overlap of the texts, which is the quantity of interest. For reporting the score, the log probability output by BARTScore is transformed into a regular probability by applying the exponential function. A large BART model[16] (M. Lewis et al., 2019) fine-tuned on the CNN Daily Mail dataset (Hermann et al., 2015) is used as the backbone model. It is the same model used in Yuan et al. (2021).

Logical Equivalence

The logical equivalence metric assesses whether the generated response and ground truth answer have the same factual content. It combines two existing methods. SummaC (Laban et al., 2022) evaluates the factual consistency of summaries by computing bi-directional entailment probabilities using an encoder model fine-tuned for NLI. Because of the limited input length of encoder models, the source and target texts are divided into sentences. The encoder is then used to compute entailment probabilities for each possible pair of source and target sentences. Because SummaC is a measure of faithfulness, the maximum entailment probabilities of the target sentences are used for computing the final score, which is their mean.

MENLI (Y. Chen & Eger, 2023) extends this precision-oriented approach to a more general evaluation of text generation via the concept of logical equivalence. While many variations are presented, Y. Chen and Eger (2023) note that computing entailment scores both from source to target and from target to source, and taking their arithmetic average is a good strategy. Rather than dividing the texts into sentences, Y. Chen and Eger (2023) compute entailment probabilities for the complete texts. However, they note that since most of the datasets used for fine-tuning models for NLI are comprised of pairs of single sentences, performance is poorer for long texts.

In this study, the following approach is taken. To accommodate for the brevity of texts in most NLI datasets as well as the limited input sizes of encoder models, the generated response and ground truth answer are split into sentences using spaCy (Honnibal et al., 2020). Max entailment probabilities are computed and averaged for both directions in order to obtain a text-level measure of semantic overlap. The final score is given by the arithmetic average of

[16] `https://huggingface.co/facebook/bart-large-cnn`

the text-level scores. The model used for computing the probabilities is a large DeBERTa V3 model[17] (He et al., 2023) trained by Laurer et al. (2024) on a variety of NLI datasets. According to the model repository, it was the best-performing NLI model on the Hugging Face Hub[18] as of June 2022.

Semantic Similarity

For computing the semantic similarity between the generated responses and ground truth answers, the approach of Akula and Garibay (2022) is adapted. Similarly to their approach, the two texts are split into sentences, and sentence embeddings are computed for each sentence. The spaCy library (Honnibal et al., 2020) was used for sentence splitting. Similarity scores are computed between each pair of generated and ground truth sentences using the cosine similarity function. Akula and Garibay (2022) then match sentences in the generated text to the most similar sentences in the reference text, and compute the average of these pairs' similarities, which functions as the final score.

However, in this study, the matching process is also done for the other direction (from reference to generated text), mirroring the approach taken with BARTScore and textual entailment. As with those metrics, the final score is given by the arithmetic average of the two scores. This has the effect of penalizing generations that do not contain information that is present in the ground truth answers, since the maximum similarity score for the corresponding ground truth sentences will be low. The model used for creating the embeddings is an MPNet (Song et al., 2020) Sentence Transformers (Reimers & Gurevych, 2019) model[19] fine-tuned on a variety of datasets for producing sentence embeddings. According to the Sentence Transformers website, this is the best of their pre-trained models (Sentence Transformers, 2025b).

3.5 Analysis

In order to assess which RAG methods affect retrieval quality, PERMANOVA (permutational multivariate analysis of variance) (Anderson, 2001) tests are conducted, using the seven evaluation metrics as dependent variables. PERMANOVA is a non-parametric alternative to the more commonly used MANOVA (multivariate analysis of variance). While PERMANOVA was originally developed for the field of ecology, it has been applied in many research areas relevant to this study, such as computational linguistics (Henriksson et al., 2024), computer-assisted language learning (Dornburg & Davin, 2024) and human-computer interaction (Chaves et al., 2019).

[17] https://huggingface.co/MoritzLaurer/DeBERTa-v3-large-mnli-fever-anli-ling-wanli
[18] https://huggingface.co/
[19] https://huggingface.co/sentence-transformers/all-mpnet-base-v2

Unlike MANOVA and ANOVA (analysis of variance), PERMANOVA operates on a matrix of pairwise distances between observations rather than the observations themselves. Assessing the significance of the test statistic is based on random permutations instead of the assumption of normality of the dependent variables, which allows PERMANOVA to be used with data that is not normally distributed. This is necessary, since preliminary analysis consisting of plotting histograms of the dependent variables reveals that BLEU and LogEq are not normally distributed.

For conducting the tests, the systems are divided into groups, which correspond to the hyperparameters that are modified (Table 3.6). Each group includes the baseline system, which uses default values for all hyperparameters. Euclidian distance matrices are computed for the observations in each group. The tests are implemented using the scikit-bio Python library[20], with the default 999 permutations and a seed value of 42. Finally, in addition to PERMANOVA, Spearman's rank correlation was employed to assess the relationship between evaluation metrics, cross-system proportion and length ratio. For each test, a significance level (α) of 0.05 is used to determine statistical significance.

[20] `https://scikit.bio/`

Chapter 4

Results

For comparing the overall performance of systems, the system-level metric scores were aggregated by standardizing and averaging them (Table 4.1). The mean Z-score of the baseline system was over three standard deviations higher than that of the non-augmented system. The difference was most notable for the two highest-scoring systems, whose Z-scores were more than one standard deviation higher than that of the baseline system.

Hyperparameter	Value	Z-score
No. of context passages	16	1.74±0.98
No. of context passages	8	1.22±0.62
No. of context passages	4	0.57±0.34
Search type	Hybrid, $k = 16$	0.42±0.33
Search type	Hybrid, $k = 2$	0.36±0.18
Re-ranking	Yes, $k = 16$	0.28±0.47
Search type	Hybrid, $k = 4$	0.26±0.24
Re-ranking	Yes, $k = 2$	0.26±0.50
Chunk overlap	320	0.25±0.53
Chunk overlap	40	0.21±0.37
Augmentation	**Yes**	0.06±0.21
Search type	Hybrid, $k = 8$	0.05±0.20

Hyperparameter	Value	Z-score
Re-ranking	Yes, $k = 8$	0.00±0.28
Chunk overlap	0	-0.01±0.30
Embedding model	Large	-0.08±0.43
Chunk overlap	80	-0.12±0.24
Chunk overlap	160	-0.20±0.35
Re-ranking	Yes, $k = 4$	-0.23±0.31
Chunk size	2048	-0.23±0.39
Search type	BM25	-0.28±0.53
Chunk size	256	-0.30±0.66
Chunk size	512	-0.54±0.66
Chunk size	128	-0.69±1.28
Augmentation	No	-3.01±1.88

Table 4.1: Means of standardized system-level metric scores with the baseline system emboldened.

The highest overall performance was obtained by the system retrieving the greatest number of context passages. The second- and third-best performing systems also belong to the group varying the number of context passages. However, the score differences between the systems are not similar for all metrics (Figure 4). The best-performing system does not obtain much higher scores than the baseline system for BLEU, ROUGE and BERTScore. The difference is moderate in BARTScore and semantic similarity (sem. sim.), and substantial in METEOR and logical equivalence (log. eq.). The full system-level evaluation results are provided in Table A.1. Histograms were computed for each measurement using the combined individual data points from all systems, and are provided in Figure B.1.

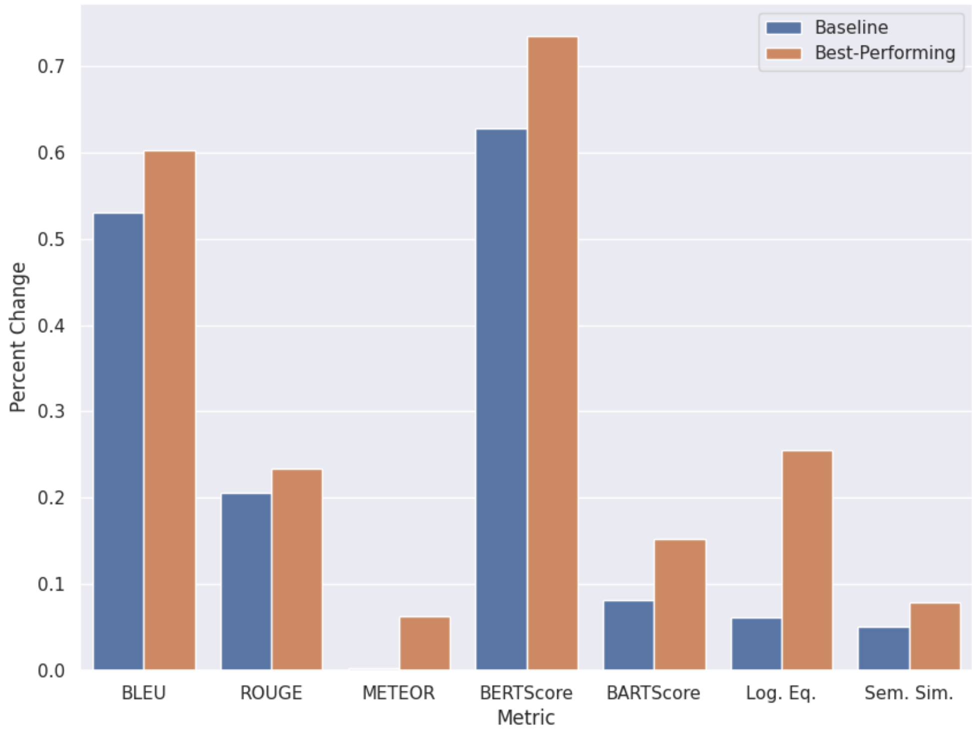

Figure 4.1: Relative improvement over the non-augmented system by the baseline and best-performing systems.

Cross-system proportion and length ratio vary little between RAG systems, but there are some notable differences with respect to the non-augmented system. The cross-system proportion scores are quite similar among the RAG systems, with the average proportion of cross-system documents among the retrieved being approximately two thirds ($M = 68.28$, $SD = 2.49$). Cross-system proportion is not measured for the non-augmented system, as this system does not retrieve context passages. The length ratios are quite different for the RAG and non-augmented systems. On average, the responses generated by RAG systems are over twice as long as the ground truth answers ($M = 262.30$, $SD = 24.09$), while those of the non-augmented system are over six times as long (Table A.1).

For each group of systems, PERMANOVA was used to test whether the corresponding hyperparameter affects performance. With $\alpha = 0.05$, the augmentation group of systems was found to be the only one where there was significant variation between systems (Table 4.2). In order to gain insight into how the individual metrics vary within each group, their coefficients of variation (CV) were computed. CV expresses the relative variability of a variable using the sample standard deviation σ and sample mean μ, and is given by

$$CV = \frac{\sigma}{\mu}. \tag{4.1}$$

Because CV is relative to the sample mean, it is independent of the unit of measurement of the variable. This allows for comparing the variability of variables that have very different value ranges, as is the case with the metric scores. In the augmentation group of systems, the CVs of BLEU, ROUGE and BERTScore are many times higher than the respective CVs in other groups (Table 4.3), while the CVs of other scores are similar to the CVs in other groups.

Hyperparameter	n	F	p
Augmentation	600	13.93	0.001
Chunk size	1,500	0.40	0.972
Chunk overlap	1,800	0.11	1.000
Embedding model	600	0.18	0.945
Search type	1,800	0.14	1.000
Re-ranking	1,500	0.18	1.000
No. of context passages	1,200	0.69	0.706

Table 4.2: Results of the PERMANOVA tests. The p values were computed using 999 permutations.

Hyperparameter	BLEU	ROUGE	METEOR	BERTScore	BARTScore	Log. eq.	Sem. sim.	Mean
Augmentation	0.30	0.13	0.00	0.34	0.05	0.04	0.04	0.13
Chunk size	0.02	0.01	0.03	0.02	0.01	0.05	0.01	0.02
Chunk overlap	0.03	0.01	0.01	0.02	0.01	0.03	0.00	0.02
Embedding model	0.02	0.00	0.00	0.00	0.00	0.00	0.01	0.00
Search type	0.02	0.00	0.01	0.01	0.01	0.03	0.01	0.01
Re-ranking	0.04	0.01	0.01	0.01	0.01	0.02	0.01	0.02
No. of context passages	0.03	0.01	0.03	0.03	0.03	0.07	0.01	0.03

Table 4.3: Within-group coefficients of variation of system-level metric scores.

In order to assess the relationships between the metric scores, cross-system proportion and length ratio, Spearman's rank correlation coefficients were computed for all pairs of measurements (Figure 4.2). Overall, the metrics have a moderately positive correlation with each other. ROUGE has the highest mean correlation coefficient with the other metrics, while logical equivalence has the lowest one (Table 4.4). None of the evaluation metrics are negatively correlated.

Cross-system proportion has a weak positive correlation with evaluation metrics. It is most correlated with BARTScore and least correlated with semantic similarity. It is also very weakly negatively correlated with length ratio. Length ratio, on the other hand, has a moderate negative correlation with ROUGE, a weak negative correlation with BLEU, BERTScore, BARTScore and semantic similarity, and a weak positive correlation with METEOR. The only metric with which it is not correlated is logical equivalence. All correlation coefficients except for the one between logical equivalence and length ratio were found statistically significant with $\alpha = 0.05$.

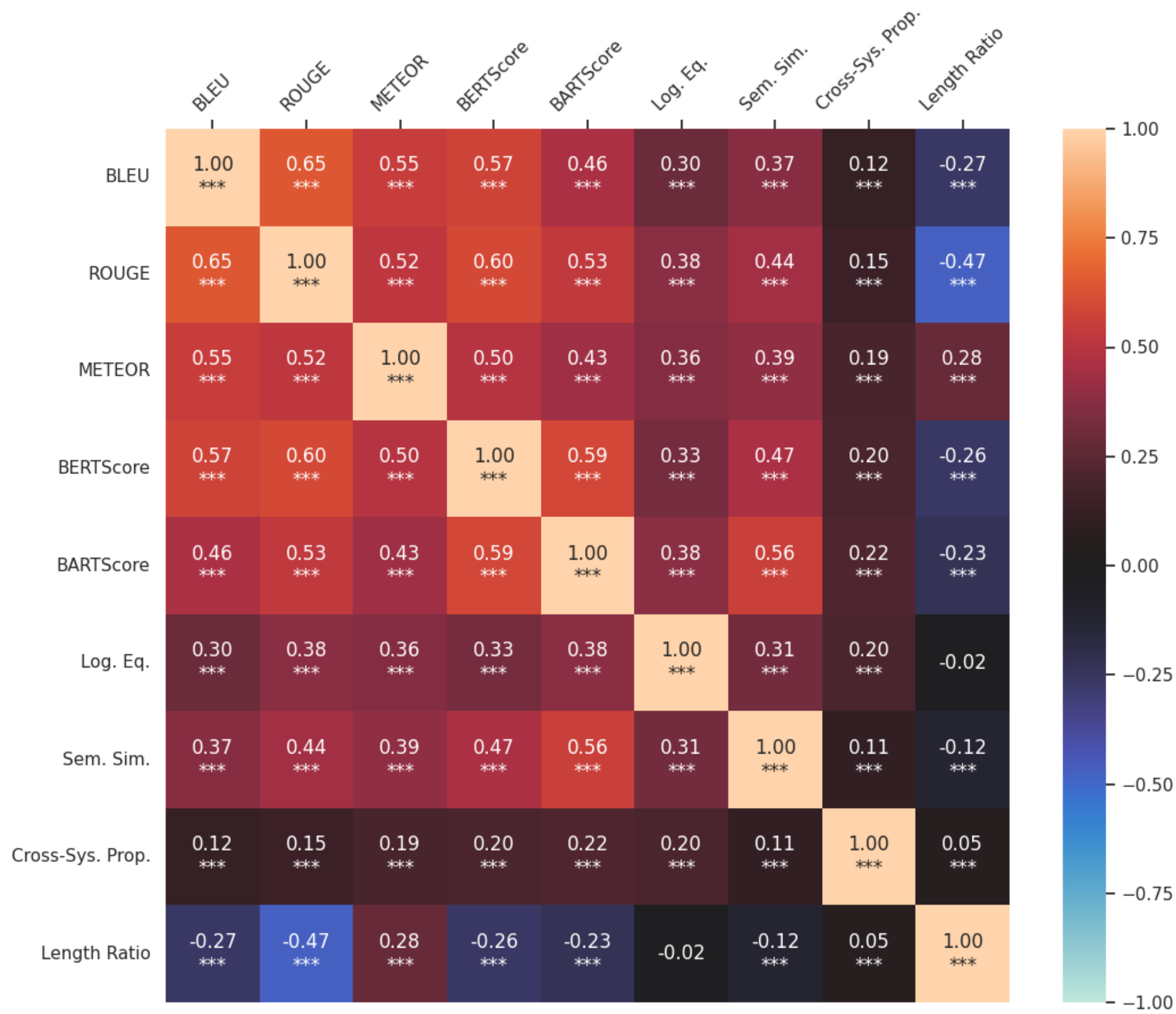

Figure 4.2: Pairwise correlation coefficients of evaluation metrics, cross-system proportion and length ratio. $*p < 0.05$. $**p < 0.01$. $***p < 0.001$.

Metric	ρ
ROUGE	0.52±0.10
BERTScore	0.51±0.10
BARTScore	0.49±0.08
BLEU	0.48±0.13
METEOR	0.46±0.07
Sem. sim.	0.43±0.08
Log. eq.	0.34±0.04

Table 4.4: Mean pairwise correlation of evaluation metrics.

Table 4.5 displays means of the metric scores, cross-system proportion and length ratio for question from each system group. All metric scores are higher for CSC questions, although the relative difference is quite low. Logical equivalence is an exception, being nearly twice as high for CSC questions compared to LUMI questions. Cross-system proportion (cross-sys. prop.) is much higher for CSC questions than LUMI questions: over 90% of the documents used for answering CSC questions belonged to the CSC group, while just over a third of the documents used for LUMI questions belonged to the LUMI groups. Length ratio is similar for the two groups, with generated responses to LUMI questions being slightly longer on average compared to the ground truth answers.

Measurement	CSC	LUMI
BLEU	2.66±2.63	2.32±2.13
ROUGE	24.58±8.73	22.35±8.96
METEOR	23.45±7.97	20.39±7.94
BERTScore	15.61±9.71	11.76±9.96
BARTScore	2.57±1.19	2.15±1.26
Logical equivalence	12.30±13.95	6.91±11.33
Semantic similarity	51.94±13.38	48.95±14.96
Cross-sys. prop.	93.82±17.64	36.21±37.75
Length ratio	255.62±387.91	304.41±723.25

Table 4.5: Average metric scores, cross-system proportion and length ratio for questions from each system group.

Passage source	CSC	LUMI	Total	No. of files
CSC				
`csc-user-guide`	8,249	4,207	12,456	612
`csc-env-eff`	2,288	594	2,882	69
Total	10,537	4,801	15,338	681
LUMI				
`lumi-userguide`	642	3,083	3,725	71
`LUMI-EasyBuild-docs`	177	1,160	1,337	4,373
Total	819	4,243	5,062	4,444

Table 4.6: Number of context passages retrieved from each data source in the experiments for each questions from each system group. The file counts from Table 3.1 are included for comparison.

Table 4.6 displays the number of context passages that were retrieved from each data source for questions from each system group. For both groups, the greatest number of passages were retrieved from sources in the CSC group, most of them from `csc-user-guide`. While for LUMI questions, the LUMI data sources were used almost as much as CSC data sources, the reverse was not true. CSC data sources were used for CSC questions over 10 times as often as LUMI data sources, while constituting a much smaller proportion of the knowledge base.

Chapter 5

Discussion

5.1 Effect of Hyperparameters

The results indicate that retrieval quality for the evaluation task is not significantly improved by modifying the RAG system. This is surprising, as previous studies conducted with an end-to-end evaluation have found optimizing the same hyperparameters critical for performance (Lyu et al., 2025; X. Wang et al., 2024). While the experiments in this study covered only a subset of possible values for each hyperparameter, it appears unlikely that experimenting with additional values would lead to different results, since greater differences between systems have been observed in previous work using a comparable setup.

In a three-document QA setting, Lyu et al. (2025) reported much higher standard deviation in system-level scores using different numbers of context passages (BLEU 0.69, ROUGE 1.06, BERTScore 0.97) than what was observed in this study (BLEU 0.07, ROUGE 0.23, BERTScore 0.40). These results were obtained using hyperparameter values within the range of $[2, 10]$, which is more limited than the one used in this study, $[2, 16]$. Considering that this hyperparameter had the highest mean coefficient of variation after augmentation, other hyperparameters are unlikely to differ in this aspect. Previous work on RAG optimization using the other evaluation metrics was not found, so the observed values of, e.g. BARTScore, could not be compared in this vein.

In order to provide insight into the low variability of the metric scores, three QA pairs are sampled for examination. The chosen pairs are the ones where there is (1) the greatest difference between non-augmented and baseline systems, (2) the greatest difference between baseline and best-performing systems and (3) the lowest average score obtained by the baseline system. For the first two, the systems' score differences for each QA pair were standardized and averaged, while for the last one, the metric scores themselves were standardized and averaged. Because of the excessive total length of the responses generated for this last pair, the pair with the second lowest average baseline score is examined instead. The pair with the lowest average score, Q274, is provided in Table C.1.

The pair with the greatest difference between non-augmented and baseline systems, Q209, demonstrates the advantage of using RAG over a non-augmented LLM (Table 5.1). The responses generated by the baseline and best-performing systems include details present in the ground truth answer, namely that on the 'standard-g' partition, 4 GPU hours are billed for each hour of node runtime and that on the 'small-g' partition, 0.5 GPU hours are billed per hour for each allocated GCD. The non-augmented response cannot provide this information. Rather, the response concerns typical policies of HPC systems, and recommends consulting system-specific ones.

Text type	Content
Question	How are GPU hours billed on standard-g and on small-g? Is it the number of GPU hours that you request for a job, using the argument #SBATCH –time, or is it the actual GPU usage per job, which is usually less than the requested hours?
Answer	For GPU compute, your project is allocated GPU-core-hours that are consumed when running jobs on the GPU nodes\nFor the standard-g partition, where full nodes are allocated, the 4 GPUs modules are billed. For the small-g and dev-g Slurm partitions, where allocation can be done at the level of Graphics Compute Dies (GCD), you will be billed at a 0.5 rate per GCD allocated.
Response	
Non-augmented	GPU hours are typically billed based on the actual GPU usage per job rather than the number of GPU hours requested using the '#SBATCH –time' argument. This means that you will be charged for the time the GPU is actively in use for your job, not for the total time you requested. Therefore, if your job finishes earlier than the requested time, you will only be billed for the actual time the GPU was utilized. Always check the specific billing policies of your computing environment, as they may vary.
Baseline	GPU hours are billed based on the number of GPUs allocated and the runtime of the job. For the 'standard-g' partitions, where full nodes are allocated, the billing is calculated as 4 GPU-hours for each hour of job runtime per node. Therefore, if you allocate 4 nodes and your job runs for 24 hours, you will be billed for 384 GPU-hours. In the case of the 'small-g' partitions, billing is done at a rate of 0.5 GPU-hour per Graphics Compute Die (GCD) allocated. If you allocate more than 8 CPU cores or more than 64 GB of memory per GCD, you will be billed per slice of 8 cores or 64 GB of memory instead. The billing is based on the allocated resources and the actual runtime of the job, rather than the requested hours specified with the '#SBATCH –time' argument.
Best-performing	GPU hours are billed based on the actual usage per job rather than the requested hours. For the 'standard-g' partition, GPU hours are billed at a rate of 4 GPU-hours for each hour of job runtime, as each node allocates 4 GPUs. In the 'small-g' partition, billing is based on the number of Graphics Compute Dies (GCDs) allocated, with a billing formula that considers the maximum of allocated GCDs, CPU cores, or memory slices, multiplied by the job runtime and adjusted by a factor of 0.5. Therefore, even if you request a longer runtime, you will only be billed for the actual GPU hours consumed during the job execution.

Table 5.1: System responses to Q209 (`lumi-1day-20230921`). This pair has the greatest score difference between non-augmented and baseline systems.

Comparing the responses generated for Q63, the QA pair with the greatest score difference between baseline and best-performing systems, provides insight into how the responses of these two systems might differ content-wise (Table 5.2). The baseline and best-performing systems provide essentially the same answer to this question, i.e. that only the manager of a computing project can modify its disk quotas. However, the best-performing system, which retrieves more passages than the baseline system, gives a less detailed answer.

Text type	Content
Question	Can only the project manager/owner apply for more disk resources into an existing workspace?
Answer	Yes, only the project manager can apply for a disk quota change.
Response	
Non-augmented	Yes, typically only the project manager or owner has the authority to apply for more disk resources in an existing workspace. This is to ensure that resource allocation is managed effectively and that any changes align with the project's goals and requirements. However, specific policies may vary depending on the organization or platform being used.
Baseline	Yes, only the project manager can modify the quotas for the scratch and projappl directories in Puhti and Mahti. For Allas, a request to increase the quota should also be sent by the project manager, specifying the project details and the required storage space.
Best-performing	Yes, only the project manager can modify the quotas and apply for more disk resources in an existing workspace.

Table 5.2: System responses to Q63 (`csc-enveff-20240207`). This pair has the greatest score difference between baseline and best-performing systems.

It is possible that the baseline system places more emphasis on details because individual phrases stand out from the limited amount of context information. In this case, the baseline system could be said to over-emphasize the details, since its response is not completely faithful to the knowledge base. While it is stated in one of the retrieved passages that only project managers are able to modify disk quotas on Puhti and Mahti, it is not explicitly mentioned whether this is also true for Allas, only that requests to increase Allas quotas should be made by email[1]. However, the baseline response states that requests for more Allas storage should also be made by the project manager.

It is noteworthy that in both of the QA pairs discussed so far, the ground truth answer is contained within each of the responses produced by the augmented systems. However, even though the non-augmented responses lack the specifics included in the former, the typical HPC practices learned from training data are enough to produce reasonable guesses. In fact, for Q63, the non-augmented response obtains higher METEOR, BARTScore and logical equivalence scores than the baseline system, while ROUGE and BERTScore are very similar (Table 5.3).

[1] https://docs.csc.fi/accounts/how-to-increase-disk-quotas/#increasing-the-storage-capacity-in-allas

Response	BLEU	ROUGE	METEOR	BERTScore	BARTScore	Log. eq.	Sem. sim.	Cross-sys. prop.	Length ratio
Q209 (`lumi-1day-20230921`)									
Non-augmented	3.59	30.19	24.34	5.79	1.48	0.33	57.93	N/A	119.05
Baseline	**10.97**	**43.32**	52.10	**31.80**	**4.14**	4.00	78.29	50.00 (1/2)	208.33
Best-performing	7.62	42.62	**53.50**	30.39	3.81	**27.94**	**78.41**	18.75 (3/16)	159.52
Q63 (`csc-enveff-20240207`)									
Non-augmented	5.41	23.53	36.73	37.77	6.39	41.25	64.75	N/A	435.71
Baseline	12.69	25.00	31.43	37.94	3.77	30.15	71.87	100.00 (2/2)	350.00
Best-performing	**32.18**	**58.06**	**73.53**	**69.08**	**9.77**	**49.81**	**88.61**	93.75 (15/16)	150.00
Q175 (`lumi-1day-20230509`)									
Non-augmented	0.00	0.00	0.00	-31.91	0.30	39.25	4.83	N/A	9800.00
Baseline	**0.70**	**4.17**	**7.69**	**-18.25**	**0.87**	8.87	**11.95**	50.00 (1/2)	5600.00
Best-performing	0.00	0.00	0.00	-29.30	0.18	**44.42**	10.13	12.50 (2/16)	7700.00
Q274 (`lumi-2day-20241210`)									
Non-augmented	0.13	2.65	5.07	-19.07	0.49	0.73	26.70	N/A	4577.78
Baseline	0.18	2.76	5.83	-13.62	1.09	1.40	10.27	100.00 (2/2)	2911.11
Best-performing	**0.79**	**3.82**	**12.43**	**-8.64**	**3.23**	**1.77**	**29.56**	68.75 (11/16)	3855.56

Table 5.3: Metric scores, cross-system proportions and length ratios of sampled responses.

RAG appears to have the greatest advantage in cases where providing exact figures or policy is useful for answering the question. Since even the system that with the highest retrieval quality delivers practically the same answers as the baseline system, it appears reasonable to conclude that the tested systems are able to answer the questions satisfactorily regardless of their configuration. The best-performing system does not offer added benefit over the baseline system, as shown both in the sampled QA pairs and the respective system-level scores (Figure 4). If differences between responses are indeed mainly surface-level ones, the low variability in scores is logical.

The fact that Lyu et al. (2025) found much higher variability in BLEU, ROUGE and BERTScore implies there was considerable variation in the relevance of retrieved passages. The apparent greater diversity in the retrieved passages of the systems of Lyu et al., 2025 could be due to the fact that their knowledge base is much larger. It contains more than 80,000 news articles, while there are only 5,125 in the one used in this study. Since the proportion of irrelevant articles for each query is greater in the knowledge base used in Lyu et al. (2025), advanced search techniques are likely more useful.

Another significant difference in the results of this study and those reported by Lyu et al., 2025 is the magnitude of system-level scores. The scores of their baseline system on the three-document question answering task (BLEU 21.05, ROUGE-L 35.04, BERTScore 87.81) are much higher than those observed in this study. This could be due to the different origins of the questions and answers that make up the evaluation datasets. While the dataset created in this study used actual user questions and expert answers, Lyu et al., 2025 used the OpenAI GPT-4 model to synthesize theirs from their corpus of news articles. Because the ground truth answers are produced by an instruction fine-tuned language model rather than individual human experts, they are likely to be more consistent in style and similar to the GPT-generated candidate responses.

Aside from Lyu et al. (2025), the two retrieval-augmented HPC QA systems built and evaluated by Yin et al. (2024) also provide an interesting point of reference. While their average ROUGE score (22.50) was similar to the ones observed in this study, their scores for BERTScore (57.00) and BARTScore (89.00) are much higher. Unlike the CRUD-RAG dataset of Lyu et al. (2025), their evaluation data was manually curated, so inherent stylistic similarities between LLM outputs and ground truth answers are unlikely to be the cause for the higher scores. However, since the data are only available by request, further analysis is not possible within the scope of this study.

The stylistic inconsistency of human answers is highlighted by the difference between Q209 (Table 5.2) and Q175 (Table 5.4). While the ground truth answer in Q209 spans three complete sentences, the one in Q175 is comprised of just one word. Q175 has the second lowest average metric score of the baseline system. Since both in Q175 and Q274 (lowest baseline average, Table C.1) the length difference between the ground truth answer and baseline answer are significant, it appears that performance on the lowest-scored QA pairs is associated with length ratio, which has a weak-to-moderate negative correlation with multiple metrics. Using a GPT model to generate a ground truth answer based on a document from the knowledge base, as Lyu et al., 2025 did, would produce an answer more similar to the candidate responses seen here. This would result in a length ratio closer to one, which could increase scores.

Text type	Content
Question	If we enable hardware threads for a job/allocation, does "–cpus-per-task" become HW threads?
Answer	Yes
Response	
Non-augmented	When you enable hardware threads for a job or allocation, the '–cpus-per-task' option typically refers to the number of hardware threads allocated to each task. In this context, if hardware threading is enabled, '–cpus-per-task' will indeed correspond to the number of hardware threads available for that task. This means that each task can utilize the specified number of hardware threads, allowing for better parallelism and potentially improved performance, depending on the workload and the architecture of the system.
Baseline	Yes, when hardware threads are enabled for a job or allocation, the '–cpus-per-task' option specifies the number of hardware threads allocated per task. This means that each task can utilize the specified number of hardware threads, allowing for better performance in applications that support multithreading.
Best-performing	Enabling hardware threads for a job or allocation means that the job can utilize the logical threads available on the CPU cores. When you specify the '–cpus-per-task' option, it can indeed refer to hardware threads, allowing each task to use multiple threads per core. This setup can enhance performance for applications that benefit from multithreading, as it allows better utilization of the CPU's capabilities.

Table 5.4: System responses to Q175 (`lumi-1day-20230509`). This pair has the second lowest average score of the baseline system. The ground truth answer is very short.

5.2 Correlation of Evaluation Metrics

The evaluation metrics are generally moderately correlated with each other, which indicates that they measure similar properties in the generated responses (Figure 4.2). Nonetheless, some metrics that had moderate positive correlation, such as ROUGE and METEOR, had very different relationships with length ratio. For example, ROUGE had a moderate negative and METEOR a weak positive correlation with length ratio. It is surprising that METEOR was positively correlated with length ratio, since other metrics computed on the level of tokens, namely BLEU, ROUGE and BERTScore, all had a weak-to-moderate negative correlation with the latter.

The correlation of metrics with length ratio is noteworthy. The system with the highest length ratio, the non-augmented one, belongs to the only group of systems where the varied hyperparameter could explain significant variation in metric scores. Although the correlation does not necessarily mean that the difference in length ratio is a significant contributor the higher score variation within the group, BLEU does explicitly penalize length differences (Papineni et al., 2002). It is possible that were the non-augmented system instructed to provide shorter responses, the result of the PERMANOVA test would not have been statistically significant for its group either.

On the other hand, greater response length can lead to generations that are less similar to the ground truth answer. For example, the non-augmented response in Table 5.2 is more verbose than the augmented responses as it addresses the question on a general level. The augmented systems may provide more concise answers as they have access to definite information regarding the topic of the question. It is thus possible that length ratio is simply positively correlated with retrieval quality, rather than the metrics being disproportionately influenced by it. This is supported by the moderate positive correlation of METEOR with other metrics despite the opposite signs of their correlation coefficients with length ratio. Nevertheless, these results highlight the known limitations of token-level metrics (Becker et al., 2024). Although used in many contemporary benchmarks, such as CRUD-RAG (Lyu et al., 2025), they may not provide an assessment of acceptable quality.

5.3 Effect of Cross-System Documents

The weak correlation coefficients for cross-system proportion (Figure 4.2) indicate that the retrieval of cross-system information is not strongly associated with lower or higher generation quality. For example, information about CSC's national HPC systems Puhti and Mahti appears to be no less useful for answering questions concerning the LUMI supercomputer than LUMI's own documentation. Although the sampled QA pairs suggested that access to detailed information is the main benefit of RAG, it seems that many questions in the evaluation dataset might benefit more from general HPC knowledge than system-specific knowledge. It is noteworthy that none of the questions in the QA pairs are strongly system-specific. However, since the computing environments of Puhti, Mahti and LUMI are very similar, it could be that RAG improves the generated responses by providing information that is detailed yet applicable across these environments.

Overall, CSC data sources appear to be more useful than LUMI data sources for answering the questions in the evaluation dataset. While there are many more LUMI documents than CSC documents in the knowledge base, the most extensive data source overall, `LUMI-EasyBuild-docs`, is also the least used (Table 4.6). This data source is also the most specialized, consisting solely of the documentation for software available on LUMI. Since the questions in the evaluation set are rather general (Tables 3.3, 3.4, 5.1, 5.2 and 5.4), knowledge of individual pieces of software is logically less useful. Conversely, the most used data source, `csc-user-guide`, was almost three times as used as the next most frequently used data source, `lumi-userguide`, and was the most used data source for questions from both system groups. The `csc-env-eff` data source was also frequently used for questions in the CSC group, which is not surprising, as this data source is comprised of the learning materials for the trainings that provided the CSC questions.

The prevalence of `csc-user-guide` is likely due to the fact that many of the documents in it also contain information on LUMI, since it is hosted by CSC. Additionally, many of the contents of `csc-user-guide` concern the basics of HPC usage, and as such are applicable on LUMI, whose computing environment is similar to that of CSC's national supercomputers. The `lumi-userguide`, on the other hand, is aimed at more advanced users. The relevance of `csc-user-guide` for the evaluation task is the likely cause of the low correlation of cross-system proportion with the evaluation metrics. Cross-sytem proportion was high for CSC and low for LUMI questions because of the preference for `csc-user-guide`, while metric scores were similar for both groups (Table 4.5). Thus, system-specificity does not appear to be especially beneficial for retrieval quality.

Chapter 6

Conclusion

This study investigated improving the retrieval quality of a RAG system for HPC question answering using reference-based end-to-end evaluation. Contrasting with previous studies on RAG optimization, (Lyu et al., 2025; X. Wang et al., 2024), it was found that tuning system hyperparameters does not significantly improve retrieval quality. While this could be based on several factors, those identified as most likely are the small size of the knowledge base as well as the inconsistent style of the ground truth answers provided by different experts, which makes it challenging to evaluate generated responses against them using text generation metrics.

In addition to this main research question, two other questions were considered. The first of these involved measuring the correlation of evaluation metrics with one another as well as response length ratio. The correlation analysis revealed that multiple metrics are at least slightly correlated with length ratio. However, it is not clear whether this reflects limitations of the metrics, or simply a relationship between length ratio and the intended evaluation target. The final research question investigated the effect of retrieving cross-system passages on retrieval quality. It was found that utilizing cross-system information is not inherently beneficial or detrimental to retrieval quality, and that especially the CSC user guide was useful for answering questions about all systems. However, while this was found to apply in the context of this particular knowledge base, it may not hold in other domains or even with the documentation of other HPC services.

For future work, one potentially rewarding avenue is the investigating how different external data sources, such as the documentation of the Slurm workload manager[1] affect retrieval quality in HPC QA. The evaluation dataset could also be extended with properly de-identified user support tickets, as Yin et al., 2024 did. Such a dataset would provided a solid basis for conducting a similar optimization study as the undertaken in this work. Finally, the use of semantic similarity and NLI-based text generation metrics for RAG evaluation could be investigated further, for example, by studying their correlation with human preferences.

[1] `https://slurm.schedmd.com/`

To the best of available knowledge, this work is the first to present a comprehensive optimization of RAG in the HPC domain, addressing a gap in existing knowledge on HPC QA. While the evaluation dataset created in this study may not be directly usable for evaluating RAG systems using other knowledge bases, the creation process documented here can serve as an example of how a similar dataset could be constructed. Additionally, the results of studying the effect of retrieving cross-system documentation provides insights into the feasibility of mixed-system knowledge bases for HPC RAG. Finally, the identification of applicable RAG evaluation metrics in previous work on text generation evaluation promotes further exploration in this area.

Bibliography

Abas, M. F. b., Singh, B., & Ahmad, K. A. (2024). High performance computing and its application in computational biomimetics. In K. A. Ahmad, N. A. W. A. Hamid, M. Jawaid, T. Khan, & B. Singh (Eds.), *High performance computing in biomimetics: Modeling, architecture and applications* (pp. 21–46). Springer Nature Singapore. https://doi.org/10.1007/978-981-97-1017-1_2

Afzal, A., Kowsik, A., Fani, R., & Matthes, F. (2024, June). Towards optimizing and evaluating a retrieval augmented QA chatbot using LLMs with human-in-the-loop. In E. Dragut, Y. Li, L. Popa, S. Vucetic, & S. Srivastava (Eds.), *Proceedings of the fifth workshop on data science with human-in-the-loop (dash 2024)* (pp. 4–16). Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.dash-1.2

Akula, R., & Garibay, I. (2022, June). Sentence pair embeddings based evaluation metric for abstractive and extractive summarization. In N. Calzolari, F. Béchet, P. Blache, K. Choukri, C. Cieri, T. Declerck, S. Goggi, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, J. Odijk, & S. Piperidis (Eds.), *Proceedings of the thirteenth language resources and evaluation conference* (pp. 6009–6017). European Language Resources Association. https://aclanthology.org/2022.lrec-1.646/

Anderson, M. J. (2001). A new method for non-parametric multivariate analysis of variance. *Austral Ecology*, *26*(1), 32–46. https://doi.org/https://doi.org/10.1111/j.1442-9993.2001.01070.pp.x

Askaryan, B. (2023). [RAG indexing diagram]. GitHub. https://github.com/langchain-ai/langchain/blob/master/docs/static/img/rag_indexing.png

Bajaj, P., Campos, D., Craswell, N., Deng, L., Gao, J., Liu, X., Majumder, R., McNamara, A., Mitra, B., Nguyen, T., Rosenberg, M., Song, X., Stoica, A., Tiwary, S., & Wang, T. (2018). *Ms marco: A human generated machine reading comprehension dataset*. arXiv: 1611.09268 `[cs.CL]`. https://arxiv.org/abs/1611.09268

Becker, J., Wahle, J. P., Gipp, B., & Ruas, T. (2024). *Text generation: A systematic literature review of tasks, evaluation, and challenges*. arXiv: 2405.15604 `[cs.CL]`. https://arxiv.org/abs/2405.15604

Biggs, B. S., & Dalton, K. (2024, May). *Outcomes of hpc user support using a science gateway ai assistant* (tech. rep.). Idaho National Laboratory (INL), Idaho Falls, ID (United States). https://doi.org/10.2172/2352719

Bird, S., Loper, E., & Klein, E. (2009). *Natural language processing with python*. O'Reilly Media Inc.

Chaves, A. P., Doerry, E., Egbert, J., & Gerosa, M. (2019). It's how you say it: Identifying appropriate register for chatbot language design. *Proceedings of the 7th International Conference on Human-Agent Interaction*, 102–109. https://doi.org/10.1145/3349537.3351901

Chen, A., Stanovsky, G., Singh, S., & Gardner, M. (2019, November). Evaluating question answering evaluation. In A. Fisch, A. Talmor, R. Jia, M. Seo, E. Choi, & D. Chen (Eds.), *Proceedings of the 2nd workshop on machine reading for question answering* (pp. 119–124). Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-5817

Chen, Y., & Eger, S. (2023). MENLI: Robust evaluation metrics from natural language inference. *Transactions of the Association for Computational Linguistics*, *11*, 804–825. https://doi.org/10.1162/tacl_a_00576

Chiang, W.-L., Li, Z., Lin, Z., Sheng, Y., Wu, Z., Zhang, H., Zheng, L., Zhuang, S., Zhuang, Y., Gonzalez, J. E., Stoica, I., & Xing, E. P. (2023, March). *Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality*. https://lmsys.org/blog/2023-03-30-vicuna/

Cormack, G. V., Clarke, C. L. A., & Buettcher, S. (2009). Reciprocal rank fusion outperforms condorcet and individual rank learning methods, 758–759. https://doi.org/10.1145/1571941.1572114

Dagan, I., Roth, D., Sammons, M., & Zanzotto, F. M. (2013). Recognizing textual entailment: Models and applications. Morgan Claypool Publishers. https://doi.org/10.1007/978-3-031-02151-0

de Lima, R. T., Gupta, S., Berrospi, C., Mishra, L., Dolfi, M., Staar, P., & Vagenas, P. (2024). *Know your rag: Dataset taxonomy and generation strategies for evaluating rag systems*. arXiv: 2411.19710 [cs.IR]. https://arxiv.org/abs/2411.19710

Ding, X., Chen, L., Emani, M., Liao, C., Lin, P.-H., Vanderbruggen, T., Xie, Z., Cerpa, A., & Du, W. (2023). Hpc-gpt: Integrating large language model for high-performance computing. *Proceedings of the SC '23 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis*, 951–960. https://doi.org/10.1145/3624062.3624172

Dongarra, J., & Keyes, D. (2024). The co-evolution of computational physics and high-performance computing. *Nature Reviews Physics*, *6*(10), 621–627. https://doi.org/10.1038/s42254-024-00750-z

Dornburg, A., & Davin, K. J. (2024). Chatgpt in foreign language lesson plan creation: Trends, variability, and historical biases. *ReCALL*, 1–16. https://doi.org/10.1017/S0958344024000272

Es, S., James, J., Espinosa Anke, L., & Schockaert, S. (2024, March). RAGAs: Automated evaluation of retrieval augmented generation. In N. Aletras & O. De Clercq (Eds.), *Proceedings of the 18th conference of the european chapter of the association for computational linguistics: System demonstrations* (pp. 150–158). Association for Computational Linguistics. https://aclanthology.org/2024.eacl-demo.16/

Ettifouri, I., Zbakh, M., & Tadonki, C. (2024). The need for hpc in ai solutions. In M. Zbakh, M. Essaaidi, C. Tadonki, A. Touhafi, & D. K. Panda (Eds.), *Artificial intelligence and high performance computing in the cloud* (pp. 137–159). Springer Nature Switzerland.

Fan, A., Jernite, Y., Perez, E., Grangier, D., Weston, J., & Auli, M. (2019, July). ELI5: Long form question answering. In A. Korhonen, D. Traum, & L. Màrquez (Eds.), *Proceedings of the 57th annual meeting of the association for computational linguistics* (pp. 3558–3567). Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1346

Friel, R., Belyi, M., & Sanyal, A. (2025). Ragbench: Explainable benchmark for retrieval-augmented generation systems. https://arxiv.org/abs/2407.11005

Gao, L., Ma, X., Lin, J., & Callan, J. (2023, July). Precise zero-shot dense retrieval without relevance labels. In A. Rogers, J. Boyd-Graber, & N. Okazaki (Eds.), *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 1762–1777). Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.acl-long.99

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., & Wang, H. (2024). Retrieval-augmented generation for large language models: A survey. https://arxiv.org/abs/2312.10997

Gehrmann, S., Clark, E., & Sellam, T. (2023). Repairing the cracked foundation: A survey of obstacles in evaluation practices for generated text. *J. Artif. Int. Res.*, *77*. https://doi.org/10.1613/jair.1.13715

Google Research. (2024). Python rouge implementation [Source code]. https://github.com/google-research/google-research/tree/master/rouge

He, P., Gao, J., & Chen, W. (2023). *Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing*. arXiv: 2111.09543 [cs.CL]. https://arxiv.org/abs/2111.09543

He, P., Liu, X., Gao, J., & Chen, W. (2021). *Deberta: Decoding-enhanced bert with disentangled attention*. arXiv: 2006.03654 [cs.CL]. https://arxiv.org/abs/2006.03654

Henriksson, E., Myntti, A., Hellström, S., Erten-Johansson, S., Eskelinen, A., Repo, L., & Laippala, V. (2024, November). From discrete to continuous classes: A situational analysis of multilingual web registers with LLM annotations. In M. Hämäläinen, E. Öhman, S. Miyagawa, K. Alnajjar, & Y. Bizzoni (Eds.), *Proceedings of the 4th international conference on natural language processing for digital humanities* (pp. 308–318). Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.nlp4dh-1.30

Hermann, K. M., Kočiský, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., & Blunsom, P. (2015). Teaching machines to read and comprehend, 1693–1701.

Honnibal, M., Montani, I., Landeghem, S. V., & Boyd, A. (2020). Spacy: Industrial-strength natural language processing in python. https://doi.org/10.5281/zenodo.1212303

Huang, D., Markovitch, D. G., & Stough, R. A. (2024). Can chatbot customer service match human service agents on customer satisfaction? an investigation in the role of trust. *Journal of Retailing and Consumer Services*, *76*, 103600. https://doi.org/https://doi.org/10.1016/j.jretconser.2023.103600

Kannagi, A., Patil, S., & Jain, S. K. (2024). Data mining techniques in bioinformatics enhanced by hpc and ai convergence. *2024 IEEE 13th International Conference on Communication Systems and Network Technologies (CSNT)*, 972–979. https://doi.org/10.1109/CSNT60213.2024.10545847

Kočiský, T., Schwarz, J., Blunsom, P., Dyer, C., Hermann, K. M., Melis, G., & Grefenstette, E. (2018). The NarrativeQA reading comprehension challenge (L. Lee, M. Johnson, K. Toutanova, & B. Roark, Eds.). *Transactions of the Association for Computational Linguistics*, *6*, 317–328. https://doi.org/10.1162/tacl_a_00023

Koh, H. Y., Ju, J., Liu, M., & Pan, S. (2022). An empirical survey on long document summarization: Datasets, models, and metrics. *ACM Comput. Surv.*, *55*(8). https://doi.org/10.1145/3545176

Krishna, K., Roy, A., & Iyyer, M. (2021, June). Hurdles to progress in long-form question answering. In K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, & Y. Zhou (Eds.), *Proceedings of the 2021 conference of the north american chapter of the association for computational linguistics: Human language technologies* (pp. 4940–4957). Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.naacl-main.393

Laban, P., Schnabel, T., Bennett, P. N., & Hearst, M. A. (2022). SummaC: Re-visiting NLI-based models for inconsistency detection in summarization (B. Roark & A. Nenkova, Eds.). *Transactions of the Association for Computational Linguistics*, *10*, 163–177. https://doi.org/10.1162/tacl_a_00453

Laurer, M., van Atteveldt, W., Casas, A., & Welbers, K. (2024). Less annotating, more classifying: Addressing the data scarcity issue of supervised machine learning with deep transfer learning and bert-nli. *Political Analysis*, *32*(1), 84–100. https://doi.org/10.1017/pan.2023.20

Lavie, A., & Agarwal, A. (2007). Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments, 228–231.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2019). BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, *abs/1910.13461*. http://arxiv.org/abs/1910.13461

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems* (pp. 9459–9474, Vol. 33). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf

Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. *Text Summarization Branches Out*, 74–81. https://aclanthology.org/W04-1013/

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., & Liang, P. (2024). Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, *12*, 157–173. https://doi.org/10.1162/tacl_a_00638

Liu, Y., Huang, L., Li, S., Chen, S., Zhou, H., Meng, F., Zhou, J., & Sun, X. (2023). Recall: A benchmark for llms robustness against external counterfactual knowledge. https://arxiv.org/abs/2311.08147

Llama Team. (2024). *The llama 3 herd of models*. arXiv: 2407.21783 [cs.AI]. https://arxiv.org/abs/2407.21783

Lù, X. H. (2024). *Bm25s: Orders of magnitude faster lexical search via eager sparse scoring*. arXiv: 2407.03618 [cs.IR]. https://arxiv.org/abs/2407.03618

Lyu, Y., Li, Z., Niu, S., Xiong, F., Tang, B., Wang, W., Wu, H., Liu, H., Xu, T., & Chen, E. (2025). Crud-rag: A comprehensive chinese benchmark for retrieval-augmented generation of large language models. *43*(2). https://doi.org/10.1145/3701228

Merkel, D. (2014). Docker: Lightweight linux containers for consistent development and deployment. *Linux Journal*, *2014*(239).

Min, B., Ross, H., Sulem, E., Veyseh, A. P. B., Nguyen, T. H., Sainz, O., Agirre, E., Heintz, I., & Roth, D. (2023). Recent advances in natural language processing via large pretrained language models: A survey. *ACM Comput. Surv.*, *56*(2). https://doi.org/10.1145/3605943

OpenAI. (2024a). *Gpt-4o system card*. arXiv: 2410.21276 [cs.CL]. https://arxiv.org/abs/2410.21276

OpenAI. (2024b). *New embedding models and api updates*. OpenAI. https://openai.com/index/new-embedding-models-and-api-updates/

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002, July). Bleu: A method for automatic evaluation of machine translation. In P. Isabelle, E. Charniak, & D. Lin (Eds.), *Proceedings of the 40th annual meeting of the association for computational linguistics* (pp. 311–318). Association for Computational Linguistics. https://doi.org/10.3115/1073083.1073135

Post, M. (2018, October). A call for clarity in reporting BLEU scores. In O. Bojar, R. Chatterjee, C. Federmann, M. Fishel, Y. Graham, B. Haddow, M. Huck, A. J. Yepes, P. Koehn, C. Monz, M. Negri, A. Névéol, M. Neves, M. Post, L. Specia, M. Turchi, & K. Verspoor (Eds.), *Proceedings of the third conference on machine translation: Research papers* (pp. 186–191). Association for Computational Linguistics. https://doi.org/10.18653/v1/W18-6319

Reimers, N., & Gurevych, I. (2019, November). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In K. Inui, J. Jiang, V. Ng, & X. Wan (Eds.), *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (emnlp-ijcnlp)* (pp. 3982–3992). Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1410

Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. M., Gatford, M., et al. (1995). Okapi at trec-3. *Nist Special Publication Sp*, *109*, 109.

Saad-Falcon, J., Khattab, O., Potts, C., & Zaharia, M. (2024, June). ARES: An automated evaluation framework for retrieval-augmented generation systems. In K. Duh, H. Gomez, & S. Bethard (Eds.), *Proceedings of the 2024 conference of the north american chapter of the association for computational linguistics: Human language technologies (volume 1: Long papers)* (pp. 338–354). Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.naacl-long.20

Şakar, T., & Emekci, H. (2025). Maximizing rag efficiency: A comparative analysis of rag methods. *Natural Language Processing*, *31*(1), 1–25. https://doi.org/10.1017/nlp.2024.53

Salemi, A., & Zamani, H. (2024). Evaluating retrieval quality in retrieval-augmented generation, 2395–2400. https://doi.org/10.1145/3626772.3657957

Scialom, T., Dray, P.-A., Lamprier, S., Piwowarski, B., Staiano, J., Wang, A., & Gallinari, P. (2021, November). QuestEval: Summarization asks for fact-based evaluation. In M.-F. Moens, X. Huang, L. Specia, & S. W.-t. Yih (Eds.), *Proceedings of the 2021 conference on empirical methods in natural language processing* (pp. 6594–6604). Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.emnlp-main.529

Sentence Transformers. (2025a, March). Pretrained models [Pretrained cross-encoder models]. https://www.sbert.net/docs/cross_encoder/pretrained_models.html

Sentence Transformers. (2025b, March). Pretrained models [Pretrained bi-encoder models]. https://www.sbert.net/docs/sentence_transformer/pretrained_models.html

Shahzad, M. F., Xu, S., An, X., & Javed, I. (2024). Assessing the impact of ai-chatbot service quality on user e-brand loyalty through chatbot user trust, experience and electronic word of mouth. *Journal of Retailing and Consumer Services*, *79*, 103867. https://doi.org/https://doi.org/10.1016/j.jretconser.2024.103867

Song, K., Tan, X., Qin, T., Lu, J., & Liu, T.-Y. (2020). Mpnet: Masked and permuted pre-training for language understanding. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems* (pp. 16857–16867, Vol. 33). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2020/file/c3a690be93aa602ee2dc0ccab5b7b67e-Paper.pdf

Su, Y., Zhou, J., Ying, J., Zhou, M., & Zhou, B. (2021). Computing infrastructure construction and optimization for high-performance computing and artificial intelligence. *CCF Transactions on High Performance Computing*, *3*(4), 331–343. https://doi.org/10.1007/s42514-021-00080-x

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., & Lample, G. (2023). *Llama: Open and efficient foundation language models*. arXiv: 2302.13971 [cs.CL]. https://arxiv.org/abs/2302.13971

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., … Scialom, T. (2023). *Llama 2: Open foundation and fine-tuned chat models*. arXiv: 2307.09288 [cs.CL]. https://arxiv.org/abs/2307.09288

TruLens. (2025, April). The rag triad. https://www.trulens.org/getting_started/core_concepts/rag_triad/

Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., & Zhou, M. (2020). *Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers*. arXiv: 2002.10957 [cs.CL]. https://arxiv.org/abs/2002.10957

Wang, X., Wang, Z., Gao, X., Zhang, F., Wu, Y., Xu, Z., Shi, T., Wang, Z., Li, S., Qian, Q., Yin, R., Lv, C., Zheng, X., & Huang, X. (2024, November). Searching for best practices in retrieval-augmented generation. In Y. Al-Onaizan, M. Bansal, & Y.-N. Chen (Eds.), *Proceedings of the 2024 conference on empirical methods in natural language processing* (pp. 17716–17736). Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.emnlp-main.981

Wei, J., Wang, X., Schuurmans, D., Bosma, M., ichter brian, b., Xia, F., Chi, E., Le, Q. V., & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, & A. Oh (Eds.), *Advances in neural information processing systems* (pp. 24824–24837, Vol. 35). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf

Williams, A., Nangia, N., & Bowman, S. (2018, June). A broad-coverage challenge corpus for sentence understanding through inference. In M. Walker, H. Ji, & A. Stent (Eds.), *Proceedings of the 2018 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long papers)* (pp. 1112–1122). Association for Computational Linguistics. https://doi.org/10.18653/v1/N18-1101

Xu, F., Song, Y., Iyyer, M., & Choi, E. (2023, July). A critical evaluation of evaluations for long-form question answering. In A. Rogers, J. Boyd-Graber, & N. Okazaki (Eds.), *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 3225–3245). Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.acl-long.181

Yin, J., Hines, J., Herron, E., Ghosal, T., Liu, H., Prentice, S., Lama, V., & Wang, F. (2024). Chathpc: Empowering hpc users with large language models. *The Journal of Supercomputing*, *81*(1). https://doi.org/10.1007/s11227-024-06637-1

Yu, H., Gan, A., Zhang, K., Tong, S., Liu, Q., & Liu, Z. (2025). Evaluation of retrieval-augmented generation: A survey. In *Big data* (pp. 102–120). Springer Nature Singapore. https://doi.org/10.1007/978-981-96-1024-2_8

Yuan, W., Neubig, G., & Liu, P. (2021). Bartscore: Evaluating generated text as text generation. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, & J. W. Vaughan (Eds.), *Advances in neural information processing systems* (pp. 27263–27277, Vol. 34). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2021/file/e4d2b6e6fdeca3e60e0f1a62fee3d9dd-Paper.pdf

Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., & Artzi, Y. (2020). Bertscore: Evaluating text generation with bert. *International Conference on Learning Representations*. https://openreview.net/forum?id=SkeHuCVFDr

Zhang, X., Yates, A., & Lin, J. (2021). Comparing score aggregation approaches for document retrieval with pretrained transformers. *Advances in Information Retrieval: 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 – April 1, 2021, Proceedings, Part II*, 150–163. https://doi.org/10.1007/978-3-030-72240-1_11

Zhao, Y., Singh, P., Bhathena, H., Ramos, B., Joshi, A., Gadiyaram, S., & Sharma, S. (2024, June). Optimizing LLM based retrieval augmented generation pipelines in the financial domain. In Y. Yang, A. Davani, A. Sil, & A. Kumar (Eds.), *Proceedings of the 2024 conference of the north american chapter of the association for computational linguistics:*

Human language technologies (volume 6: Industry track) (pp. 279–294). Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.naacl-industry.23

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., Zhang, H., Gonzalez, J. E., & Stoica, I. (2023). Judging llm-as-a-judge with mt-bench and chatbot arena. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, & S. Levine (Eds.), *Advances in neural information processing systems* (pp. 46595–46623, Vol. 36). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2023/file/91f18a1287b398d378ef22505bf41832-Paper-Datasets_and_Benchmarks.pdf

Zhu, K., Luo, Y., Xu, D., Yan, Y., Liu, Z., Yu, S., Wang, R., Wang, S., Li, Y., Zhang, N., Han, X., Liu, Z., & Sun, M. (2025). *Rageval: Scenario specific rag evaluation dataset generation framework*. arXiv: 2408.01262 [cs.CL]. https://arxiv.org/abs/2408.01262

Appendix A

Evaluation Results

Parameter value	BLEU	ROUGE	METEOR	BERTScore	BARTScore	Log. eq.	Sem. sim.	Cross-sys. prop.	Length ratio
Augmentation									
No	1.64±1.40	19.61±8.85	22.04±6.88	8.60±9.69	2.19±1.03	9.40±11.54	48.42±12.93	N/A	621.04±1101.66
<u>Yes</u>	**2.51±2.04**	**23.64±8.76**	**22.09±7.68**	**14.01±9.67**	**2.37±1.13**	**9.98±13.55**	**50.88±13.88**	69.67±40.02	266.92±474.70
Chunk size									
128	**2.59±2.81**	23.39±8.72	20.45±8.65	**14.36±10.12**	**2.37±1.21**	8.83±12.11	50.36±14.61	68.00±42.61	198.13±427.60
256	2.58±2.60	**23.84±8.80**	21.31±8.37	14.14±9.94	2.36±1.18	9.43±13.26	50.39±14.94	69.33±42.22	228.41±444.48
512	2.50±2.07	23.70±8.61	21.73±7.86	14.17±9.71	2.29±1.07	9.15±12.17	50.06±14.21	69.67±41.26	234.32±472.67
<u>1024</u>	2.51±2.04	23.64±8.76	22.09±7.68	14.01±9.67	**2.37±1.13**	**9.98±13.55**	**50.88±13.88**	69.67±40.02	266.92±474.70
2048	2.50±2.20	23.73±8.93	**22.11±8.12**	13.78±10.21	2.35±1.15	9.32±12.08	50.50±14.22	70.33±40.11	278.00±557.43
Chunk overlap									
0	2.53±2.04	23.79±8.70	21.91±7.84	14.00±9.97	2.37±1.13	9.72±13.32	50.87±14.38	69.67±40.44	260.90±523.91
<u>20</u>	2.51±2.04	23.64±8.76	22.09±7.68	14.01±9.67	2.37±1.13	9.98±13.55	50.88±13.88	69.67±40.02	266.92±474.70
40	2.56±2.70	23.81±8.70	**22.33±7.96**	13.99±9.95	2.35±1.13	**10.08±12.46**	**51.02±14.20**	69.50±40.42	256.95±440.62
80	2.46±2.01	23.45±8.74	21.91±7.72	13.98±9.77	2.38±1.16	9.64±13.11	50.78±14.10	70.00±40.89	276.19±605.01
160	2.50±2.60	23.61±8.80	21.91±8.30	14.35±10.34	2.36±1.20	9.47±12.65	50.37±14.34	70.83±40.99	268.22±571.21
320	**2.67±3.27**	**23.91±9.20**	22.16±8.26	**14.48±10.37**	**2.43±1.26**	9.40±12.80	50.64±14.39	71.17±41.42	258.93±453.80
Embedding model									
<u>Small</u>	2.51±2.04	23.64±8.76	**22.09±7.68**	**14.01±9.67**	**2.37±1.13**	9.98±13.55	**50.88±13.88**	69.67±40.02	266.92±474.70
Large	**2.57±2.11**	**23.77±8.94**	22.05±8.21	13.95±9.93	2.36±1.07	**10.03±12.73**	50.04±14.39	71.67±38.32	260.07±510.51
Search type									
<u>Semantic</u>	2.51±2.04	23.64±8.76	22.09±7.68	14.01±9.67	2.37±1.13	9.98±13.55	**50.88±13.88**	69.67±40.02	266.92±474.70
BM25	2.44±2.10	23.58±8.53	21.62±7.96	13.79±9.32	2.43±1.35	9.54±12.56	50.15±13.70	62.67±42.57	238.58±420.29
Hybrid, $k = 2$	**2.58±2.71**	23.78±8.74	22.23±8.30	14.07±10.13	2.43±1.37	10.28±13.60	50.80±14.34	66.00±41.02	264.67±537.13
Hybrid, $k = 4$	2.51±2.68	**23.79±8.80**	22.36±8.54	13.97±10.17	2.43±1.41	9.97±13.16	50.80±13.70	66.33±41.50	272.16±609.84
Hybrid, $k = 8$	2.49±2.68	23.58±8.86	22.12±8.42	13.90±9.97	2.38±1.34	10.22±13.67	50.61±14.29	66.83±40.79	258.31±492.14
Hybrid, $k = 16$	2.56±2.76	23.73±9.13	**22.37±8.67**	**14.18±10.23**	**2.44±1.47**	**10.51±14.58**	50.64±14.41	66.83±41.20	264.28±534.39
Re-ranking									
<u>No</u>	2.51±2.04	23.64±8.76	22.09±7.68	14.01±9.67	2.37±1.13	9.98±13.55	50.88±13.88	66.83±41.20	266.92±474.70
Yes, $k = 2$	2.51±2.05	**24.18±8.50**	22.25±7.72	**14.38±9.79**	2.36±1.08	9.73±12.15	**51.26±13.80**	69.67±40.02	247.62±447.21
Yes, $k = 4$	2.47±2.11	23.74±8.65	22.12±8.37	13.83±9.89	2.34±1.14	9.66±12.86	50.28±14.33	69.83±39.84	267.76±545.29
Yes, $k = 8$	2.46±2.18	23.53±8.88	21.89±7.78	13.94±9.36	2.39±1.40	**10.22±13.56**	50.69±14.84	69.00±39.72	264.65±488.38
Yes, $k = 16$	**2.68±2.65**	23.91±9.28	**22.48±8.20**	14.07±9.49	**2.41±1.42**	9.75±13.13	50.38±14.33	67.83±40.36	259.17±467.81
No. of context passages									
<u>2</u>	2.51±2.04	23.64±8.76	22.09±7.68	14.01±9.67	2.37±1.13	9.98±13.55	50.88±13.88	69.67±40.02	266.92±474.70
4	2.56±2.64	23.83±9.08	22.40±8.01	14.21±10.16	2.45±1.27	10.59±13.97	50.97±13.90	67.92±37.89	292.21±614.16
8	**2.67±2.71**	24.02±9.19	22.91±8.19	14.43±10.55	**2.52±1.41**	11.09±14.36	51.48±14.28	65.50±36.22	300.94±632.01
16	2.63±2.70	**24.18±9.65**	**23.42±8.52**	**14.94±10.50**	**2.52±1.22**	**11.80±14.79**	**52.18±14.14**	62.17±35.98	315.59±632.11

Table A.1: System-level metric scores, cross-system proportion and length ratio. Baseline parameter values are underlined and the highest metric scores for each parameter are emboldened. Cross-system proportion is not reported for the non-augmented system, since it does not retrieve context passages.

Appendix B

Histograms of Measurements

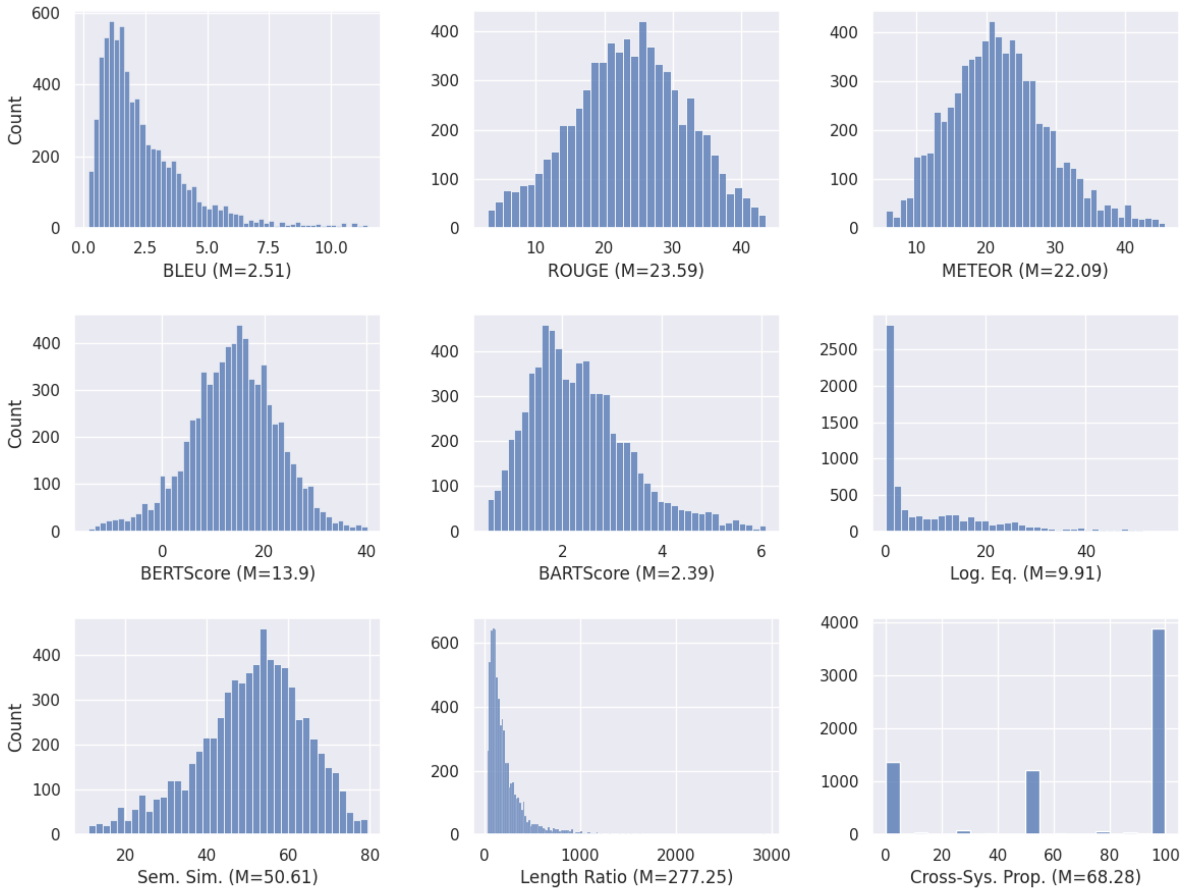

Figure B.1: Histograms of the evaluation metrics, cross-system proportion and length ratio. The data bins were computed using the combined individual data points from all systems.

Appendix C

QA Pair With Lowest Baseline Z-Score

Text type	Content
Question	How can I find the tasks in LUMI that have been finished/calculated and ran in the past two days/24h? Can you give me some tips and guidance?
Answer	The ‘sacct‘ command is your friend.
Response	
Non-augmented	To find tasks in LUMI that have been finished or calculated and ran in the past two days (or 24 hours), you can follow these steps:\n\n1. **Access the LUMI System**: Log in to the LUMI system using your credentials.\n\n2. **Navigate to the Job Management Interface**: Look for a section related to job or task management. This is often labeled as "Jobs," "Tasks," or "Job History."\n\n3. **Filter by Status**: Use the filtering options to select tasks that have a status indicating they are finished or completed. This may be labeled as "Completed," "Finished," or similar.\n\n4. **Set Time Filters**: Look for options to filter tasks by time. You should be able to set a custom date range or select a predefined range (like "Last 24 hours" or "Last 2 days"). If there is a date picker, select the appropriate dates.\n\n5. **Sort or Search**: If the interface allows, sort the results by the submission or completion time to easily identify the most recent tasks.\n\n6. **Review Task Details**: Click on individual tasks to view more details, such as the output, logs, and any relevant metrics.\n\n7. **Use Command Line Tools (if applicable)**: If LUMI provides a command line interface (CLI), you can use commands to query job statuses and filter by time. Refer to the documentation for specific commands.\n\n8. **Check Documentation**: If you are unsure about the filtering options or how to navigate the interface, consult the LUMI user documentation or help resources for guidance.\n\n9. **Contact Support**: If you encounter any issues or need further assistance, consider reaching out to LUMI support or your institution’s IT helpdesk.\n\nBy following these steps, you should be able to efficiently find the tasks that have been finished or calculated in the last two days.
Baseline	To find tasks that have been finished or calculated in LUMI over the past two days or 24 hours, you can utilize the job scheduling and monitoring tools available on the system. Here are some tips and guidance:\n\n1. **Check Job Status**: Use the job status command to view the history of your jobs. This will typically show you completed jobs along with their status.\n\n2. **Job Logs**: Review the logs generated by your jobs. These logs often contain timestamps and can help you identify when tasks were completed.\n\n3. **Resource Management Commands**: While specific commands for retrieving historical job data are not mentioned, you may want to explore any available commands related to job history or completed tasks in the LUMI environment.\n\n4. **Contact Support**: If you are unable to find the information you need, consider reaching out to the LUMI user support team for assistance. They may have access to more detailed records or tools that can help you track your completed tasks.\n\n5. **Documentation Review**: Look through the user documentation for any additional commands or features that may assist in tracking job completion and history.\n\nBy following these steps, you should be able to gather information about the tasks that have been completed in the last 24 hours.
Best-performing	To find tasks that have been completed in LUMI over the past two days or 24 hours, you can utilize the web interface or command-line tools available on LUMI.\n\n1. **Using the Web Interface:**\n - After logging into the LUMI web interface, navigate to the "My Interactive Sessions" page. This section will display your currently running and past interactive app sessions.\n - You can also check the status of your batch jobs by looking for the "Active Jobs" app under the "Jobs" section in the top navbar. This will show you details about your recently completed Slurm jobs.\n\n2. **Using Command-Line Tools:**\n - If you prefer using the command line, you can use the ‘sacct‘ command, which is part of the Slurm job scheduler. This command allows you to view accounting data for jobs that have completed.\n - You can run a command like:\n ‘‘‘bash\n sacct –starttime=YYYY-MM-DDTHH:MM:SS –format=JobID,JobName,State,Elapsed\n ‘‘‘\n Replace ‘YYYY-MM-DDTHH:MM:SS‘ with the appropriate timestamp for 24 hours ago. This will list jobs that have finished within the specified time frame, along with their job ID, name, state, and elapsed time.\n\n3. **Tips:**\n - Make sure to check the job state to confirm that they have completed successfully.\n - If you need more detailed information about specific jobs, you can use the job ID with the ‘scontrol‘ command to get additional details.\n\nBy following these steps, you should be able to effectively track and review the tasks that have been completed in LUMI over the past two days.

Table C.1: System responses to Q274 (`lumi-2day-20241210`).