CHALMERS
UNIVERSITY OF TECHNOLOGY

Using LoRA to Improve Retrieval in Structured Document Collections

An Empirical Study of LoRA Roles and Rank Scaling in Retrieval-Augmented Generation

Master's thesis in Data Science & AI

Marcus Wassenius
Junayd Kader

DEPARTMENT OF MATHEMATICAL SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

Using LoRA to Improve Retrieval in Structured Document Collections

An Empirical Study of LoRA Roles and Rank Scaling in Retrieval-Augmented Generation

MARCUS WASSENIUS
JUNAYD KADER

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Division of Analysis and Probability Theory
Chalmers University of Technology
Gothenburg, Sweden 2026

Using LoRA to Improve Retrieval in Structured Document Collections
An Empirical Study of LoRA Roles and Rank Scaling in Retrieval-Augmented Generation
MARCUS WASSENIUS
JUNAYD KADER

Supervisor: Sergei Zuyev, Mathematical Sciences
Examiner: Sergei Zuyev, Mathematical Sciences

Master's Thesis 2026
Department of Mathematical Sciences
Division of Analysis and Probability Theory
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Illustration of a network.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

Using LoRA to Improve Retrieval in Structured Document Collections
An Empirical Study of LoRA Roles and Rank Scaling in
Retrieval-Augmented Generation
MARCUS WASSENIUS
JUNAYD KADER
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

This thesis explores different roles for Low-Rank Adaptation (LoRA) in a question answering (QA) system over structured document collections, and aims to shed light on benefits and limitations as well as rank requirement interactions with dataset size and confusability. Three roles are investigated on synthetic document collections mimicking company data. First, LoRA as parametric memory to internalize new knowledge directly into the model weights; second, LoRA as a tool for embedding adaptation in a retrieval-augmented generation (RAG) system to improve retrieval performance by improving the vector representations of chunks; third, LoRA as a tool for context routing in a RAG system to improve retrieval by narrowing the search space.

The findings suggest that LoRA is a flexible tool that can improve QA performance in all three roles, but that each role comes with its own characteristics and trade-offs between for example performance, flexibility, and robustness. In the specific settings tested, parametric memory is shown to be effective and highly rank sensitive; embedding adaptation is shown to give robust retrieval and generation improvements already at low ranks with gradual benefits of increased rank; context routing is shown to give large retrieval benefits at low rank requirements but introduces certain brittleness in both the retrieval and generation step.

Keywords: Low-Rank Adaptation (LoRA), parameter-efficient fine-tuning (PEFT), Retrieval-Augmented Generation (RAG), embedding adaptation, rank scaling, structured document retrieval, context routing.

Acknowledgements

We would like to express our gratitude to Sergei Zuyev for taking us on as both supervisor and examiner at a time when the thesis process was uncertain and stressful. His confidence in us and in the project gave us important encouragement, and his guidance has been highly valuable throughout the project. We are particularly grateful for the generously long, always insightful, and intellectually challenging conversations in his office, which provided us with both direction and inspiration.

We would also like to thank Konsert Strategy & IP for their support throughout the project. Their contribution of computational resources made the experimental work possible, and they provided us with valuable insights on how language models, retrieval, and LoRA are being explored in consulting contexts which has not only been insightful for us personally, but also gave the thesis a valuable practical dimension.

Marcus Wassenius and Junayd Kader, Gothenburg, May 2026

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis, listed in alphabetical order:

AI	Artificial Intelligence
BERT	Bidirectional Encoder Representations from Transformers
FAISS	Facebook AI Similarity Search
GIP	Gold-in-Prediction
GPU	Graphics Processing Unit
LLM	Large Language Model
LoRA	Low-Rank Adaptation
MI	Mutual Information
MRR	Mean Reciprocal Rank
NF4	NormalFloat 4-bit
NLP	Natural Language Processing
PEFT	Parameter-Efficient Fine-Tuning
PRAG	Parametric Retrieval-Augmented Generation
QA	Question Answering
QLoRA	Quantized Low-Rank Adaptation
RAG	Retrieval-Augmented Generation
SVD	Singular Value Decomposition

Contents

List of Figures

List of Tables

1
Introduction

1.1 Background

Large language models (LLMs) are increasingly used in settings that require access to structured facts, such as enterprise document analysis, technical documentation, and knowledge management systems [43]. In many such settings, information is not stored as unstructured text but instead exists in collections of documents that follow common templates (such as reports with sections and subsections) and that contain facts in a structured manner. Enabling LLMs to accurately make use of such information has therefore become an important research problem.

One widely used approach for handling knowledge-intensive tasks is Retrieval-Augmented Generation (RAG), where a language model retrieves relevant information from an external knowledge corpus before generating an answer to the user's query [23]. This is commonly done by embedding documents into a vector space, upon which the embeddings closest to the user query's embedding are returned to the language model as context. This approach allows models to access knowledge that was not present during the pre-training.

An alternative approach is to incorporate the external knowledge directly into the parameters of a language model through fine-tuning - however, fully fine-tuning modern LLMs is computationally expensive due to the large number of parameters involved, which has motivated the development of *parameter-efficient fine-tuning* (PEFT) methods such as Low-Rank Adaptation (LoRA) [8, 14]. Instead of updating the full parameter space, LoRA learns a low-dimensional update that modifies the model's behaviour while keeping the base model frozen. The capacity of LoRA adapters is controlled, to a large extent, by a parameter known as the *rank*.

While RAG and LoRA are fundamentally different approaches to solving similar problems - making use of new knowledge - there have been some recent studies that have explored ways of combining LoRA with retrieval-based systems. For example, LoRA can be applied to the embedding models to improve retrieval [5], or used after retrieval in the form of LoRA-adapters associated with the retrieved documents [44].

Despite these developments, several questions remain open. In particular, it is to the best of our knowledge not yet well understood how the effectiveness of LoRA depends on where it is applied within a question-answering pipeline or how the

required LoRA rank scales as the difficulty of the retrieval task increases. When document collections grow larger or become more confusable, it seems likely that retrieval becomes more challenging considering the way RAG systems function, and it is unclear how well LoRA can compensate for this difficulty and how much adapter capacity is required for this.

To investigate these questions in a controlled way, this thesis studies the use of LoRA within synthetic, structured document collections designed to mimic enterprise-style documents. By systematically varying properties of the dataset, such as the number of contexts and the confusability between documents, the study aims to provide understanding regarding both the benefits and limitations of different LoRA applications within a question-answering system and how LoRA capacity requirements (in terms of rank) change across the different applications as the task becomes more difficult. This study aims to provide empirical insight into the role of parameter-efficient fine-tuning in retrieval-based language model systems and also seeks to contribute to a better understanding of how LoRA can be applied effectively in knowledge-intensive settings.

1.2 Aim

The aim of this thesis is to investigate how Low-Rank Adaptation (LoRA) can be used to improve question answering over structured document collections and to determine how the required LoRA capacity changes as the knowledge base becomes more challenging in terms of size and confusability.

In particular, the work focuses on understanding two key aspects of LoRA usage in retrieval systems. First, the thesis studies where in a question-answering pipeline LoRA gives the most benefit. LoRA can potentially be applied in several different roles as shown by prior work, including storing knowledge directly in the model parameters (through the LoRA adapters), adapting the embedding space used for retrieval, or potentially by guiding retrieval toward the most relevant context before searching the corpus. Understanding which of these roles is most effective is important for designing efficient systems for LLMs to make use of new knowledge, and the effectiveness and implications of these three roles are central to this thesis.

Second, the thesis studies how the LoRA rank required for strong performance changes as the difficulty of the underlying knowledge base increases, and how it differs between the different applications of LoRA. The difficulty of the task is controlled by systematically varying properties of the document collection, including the number of contexts in the knowledge base and the degree that contexts are confusable with each other. These factors affect how challenging it is for knowledge internalization using LoRA, and how challenging it is for a retrieval system to identify the correct information among many similar alternatives.

More specifically, the work aims to:

- compare different ways of applying LoRA in retrieval-based question answering

systems, including LoRA as parametric memory, LoRA-adapted embedding retrieval, and LoRA-based context routing;
- study how the performance of these systems changes as the size and difficulty of the knowledge base increase;
- estimate, within a fixed rank sweep, fixed epoch budget, and fixed hyperparameter setup, the minimum LoRA that appears sufficient under different levels of difficulty;
- provide empirical insight into how LoRA capacity requirements for these different applications scale as the knowledge base grows in size and complexity.

1.3 Research Questions

The thesis is built around two main research questions:

1. What are the benefits and limitations of LoRA when applied in a question-answering system as parametric memory, when used for retrieval adaptation, and when used for context routing?
2. How does the required LoRA rank change in these three roles, as the knowledge base becomes more challenging?

These questions can be further decomposed into more specific sub-questions:

- How well can LoRA internalize structured knowledge directly in model weights?
- How well can LoRA improve retrieval by adapting the embedding model used for document retrieval?
- How well can LoRA improve retrieval by guiding the system toward the correct context before performing retrieval?
- How do LoRA rank requirements change as the number of contexts grows and as documents become more difficult to discriminate between, across these different applications?

To study these questions in a controlled way, the document corpus used in the experiments can be described by two main variables **N** and **S**, where N is the number of contexts in the dataset, and S is the level of confusability between contexts.

These parameters allow the difficulty of the experiments to be systematically controlled. Increasing N increases the number of possible locations where relevant information may be located and simultaneously the number of individual facts, while increasing S makes documents more confusable by making them more similar in terms of both vocabulary and semantic themes and topics, while also making the wordings of facts less explicit, which all-in-all should make embedding-based retrieval more difficult.

1.4 Scope and Limitations

This thesis is scoped to study the role and capacity of LoRA in a controlled, synthetic setting, and several limitations follow from this design choice which should be considered when interpreting the results.

First, the document collections used in the experiments are synthetically generated rather than taken from real-world datasets. The synthetic document corpus allows the structure of the dataset and the difficulty of the retrieval problem to be controlled and varied, for example by varying the number of contexts or the confusability between them. However real-world documents can often contain additional complexities, such as inconsistent formatting, incomplete information, non-informative metadata, and other aspects that are not replicated by this study and that could in theory affect the complexity of the tasks we are evaluating the systems on. As a result, the findings of this thesis should be interpreted primarily as empirical evidence on controlled datasets with the specific attributes we describe, rather than as a direct representation of all real-world settings.

Second, the experiments are conducted under fixed experimental conditions, including specific system architectures, a specific embedding model, and specific LoRA and RAG configurations. For example, model sizes, chunking approaches, or alternative parameter-efficient fine-tuning methods may exhibit different scaling behaviour and different retrieval and generation performance. The conclusions should therefore be understood as specific to the experimental setup used in this thesis.

Third, the study fixes the number of epochs across all experiments. This was done particularly with the system testing parametric memory in mind, in order to ensure that as data size grows, the model is exposed to each fact the same number of times (if freezing the number of optimization steps instead, each fact would get less and less exposure). Thus larger datasets get more steps. Furthermore, direct comparisons across systems should be done with care: the parametric-memory system is exposed to each fact in two different query formulations in training, doubling the training set size and thus the number of optimization steps compared to the other systems which are not trained on direct fact memorization and which are only trained on one training instance per piece of knowledge. Although all systems converge, this still introduces a confound when comparing the systems in terms of absolute performance gains. The study should therefore not be read as a comparison of each system under an equal number of optimization steps.

Fourth, the study isolates only a limited set of factors that may influence retrieval difficulty. In particular, the experiments vary the number of contexts and the confusability between contexts. While these variables capture important aspects of retrieval difficulty, other factors such as document length variation, domain differences, or noisy data are not considered within the scope of this study.

Fifth, the study's findings are based on limited seed runs, in most cases three seeds per experiment configuration. The results should therefore be interpreted as observed empirical patterns rather than statistically significant scaling laws.

Finally, the evaluation metrics focus primarily on structured question answering accuracy and retrieval measures. While these metrics provide clear and interpretable indications on the performance of the systems in our experiments, they may not

capture all aspects of model usefulness in real-world applications, such as the quality of longer explanations or open-ended questions.

Despite these limitations, the controlled experimental setup allows the thesis to isolate the effects of LoRA and its rank on retrieval performance, and is believed to enable experiments that may help answer the research questions.

2
Theory

2.1 Background on Language Models

This section provides a brief overview of language models and their usage of knowledge. In particular, this section describes the transformer architecture as well as how knowledge can be stored both within a model's parameters and accessed from outside the model.

2.1.1 Transformer Language Models

Language models today are often based on the transformer architecture [19]. A language model generates text by predicting one token at a time based on text that has already been generated previously. In a transformer model the input text is first split into tokens and each token is then turned into a numerical representation which is called an embedding. After that, information about the position in the sequence of each token is added. The embeddings are passed through multiple layers called transformer blocks, that together refine the embedding of each token. An important aspect of these transformer blocks is the self-attention mechanism. This allows the model to determine which parts of the input are most important when processing each token which enables the model to capture relationships between words even when they are far apart in the text. After the final layer (transformer block), the model gives a probability distribution over next possible tokens, and the most likely token is selected as the output.

A language model's outputs depend on the knowledge it has acquired during training along with any information it has access to at inference time. There is thus a clear distinction between different ways knowledge can be represented and used in language models.

2.1.2 Knowledge in Language Models

Overall the knowledge used by a language model can be seen as coming from two types of sources: parametric knowledge, and non-parametric knowledge.

Parametric knowledge is the knowledge stored in the model's parameters. During training, language models are exposed to large amounts of text data, which is knowledge that becomes encoded into the weights during training. Petroni et al. (2019) show that pretrained models can answer fact questions in a way that indicates

that the models store relational knowledge internally (within their parameters) [31]. Their work also suggests that certain types of facts, such as more simple factual relationships, are easier to encode and retrieve in order to answer fact questions than other types of facts, such as more complex relationships. This in turn suggests that knowledge storage in language models depends on factors such as relational structure of the data.

Building on this, Roberts et al. (2020) investigate how much knowledge can be stored in model parameters, and they find that performance improves with model size [37]. As such it seems that larger models are more capable of storing and retrieving facts.

However not all knowledge needs to be stored inside the model parameters. Non-parametric knowledge refers to, in this thesis, information that is stored in an external source and accessed at inference time instead of being encoded into the model's parameters during training. This external knowledge may consist of, for example, documents, that are retrieved at inference time to help answer the question [23].

2.2 Knowledge Storage and Access

This section expands on the distinction between parametric and non-parametric knowledge by describing two major approaches for how language models can use information. In particular, this section covers PEFT methods with a particular emphasis on LoRA, as well as RAG.

2.2.1 Parameter-Efficient Fine-Tuning and Low-Rank Adaptation

In terms of parametric knowledge, fully fine-tuning a large language model is often computationally expensive since in order to fully fine-tune, all the model's parameters must be updated [8]. This means high memory costs and computational costs. Parameter-efficient fine-tuning (PEFT) methods are methods that address this issue by keeping most of the original base model parameters frozen while updating only a small number of additional parameters instead.

One PEFT method is called Low-Rank Adaptation (LoRA), developed by researchers at Microsoft in 2021 [14]. LoRA works by learning a set of low-rank update weight matrices that are added to the frozen weight matrices of the base model. This way, the low-rank update adapts the model behavior in an efficient way. More formally, let $W_0 \in \mathbb{R}^{d \times k}$ be a pretrained weight matrix of some choice of base model. If fine-tuning normally, this matrix would be fully updated during training. However, with LoRA, this matrix W_0 is frozen and the updates are instead done on two new matrices with low rank B and A:

$$W = W_0 + \Delta W, \qquad \Delta W = BA,$$

$B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$ are the trainable matrices that together, through matrix multiplication, form ΔW that has the same size as W, and the rank r is chosen in such a way that $r \ll \min(d, k)$ [14]. So therefore the total number of trainable parameters is much smaller than in full fine-tuning.

LoRA has many advantages. Since only a fraction of the number of parameters is trained, computational requirements when using LoRA are much less compared to when using full fine-tuning. Another advantage is that task-specific LoRA adapters can efficiently be stored due to the relatively small size of B and A, and the low-rank update ΔW can be integrated into the original base model matrix to minimize answer latency. In practice, LoRA is often used together with quantization techniques to further reduce memory usage during the fine-tuning process - this is often done with so-called QLoRA [7]. In QLoRA, the frozen base model's weights are stored in 4-bit precision (usually with the NF4 data type) which means that each weight is represented with far fewer bits than when using for example 16-bit precision. Moreover, so-called double quantization which quantizes the quantization constants, and paged optimizers which help with temporary memory spikes, are used in QLoRA which altogether make LoRA fine-tuning more feasible on limited GPU hardware.

The effectiveness of LoRA depends on modeling choices such as which base model layers to apply low-rank updates to, and what rank should be used (which dictates the size of the trainable matrices A and B): thus users must be aware of how design choices influence the power of LoRA [14, 8].

2.2.2 Retrieval-Augmented Generation

In terms of non-parametric knowledge, a technique called Retrieval-Augmented Generation combines it with the parametric knowledge of a language model to make use of external knowledge [23, 26]. At a very high level, a RAG system consists of two main components: a retriever step and a generator step. When using RAG, documents are typically divided into smaller parts or "chunks", embedded into a vector space, and stored. When a query is given by a user, the system retrieves the most relevant chunks (where relevance is determined by comparing the embedded query to the chunk embeddings) and gives them to the language model as "context" for generation. Then, the generator creates an output that is conditional on both the user query and the retrieved chunks. So in basic terms, retrieval acts as a mechanism for providing the base model with the factual context needed to answer a query.

A big advantage of RAG is that knowledge is stored in the external database and can be updated or replaced easily, which is not the case with parametric knowledge since when the knowledge changes, the weights need to be updated through training. Another aspect is the fact that the documents retrieved can be inspected and thus allows good interpretability in terms of why the system performs the way it does.

At the same time, RAG systems hold trade-offs. For example, retrieval-based systems need access to an external knowledge source at inference time [37]. Moreover,

since retrieval is based on embedding similarity [26], the performance of the retrieval may become more challenging when document collections contain sections with similar language or semantic content, making chunks that may be related to the wrong true fact still *appear* relevant enough for retrieval due to the embedding similarity. Additionally, RAG systems introduce architectural complexity and may introduce additional latency or security risks [11, 28].

2.2.3 Hybrid Parametric-Retrieval Approaches

Although parametric adaptation with methods such as PEFT and retrieval-based systems such as RAG are quite distinct approaches, there has been recent work on how the two can be combined. These hybrid approaches aim to capture the benefits of retrieval methods while improving some parts of the system pipeline with PEFT. So rather than treating parametric and non-parametric knowledge as fully separate regimes, this line of work involving hybrid approaches investigates how they can be used together to improve the overall performance of a system.

One hybrid approach is to apply LoRA to the retrieval step, as described in the work by Choi et al. [5]. In their study, LoRA is applied to a BERT-based retriever, and documents are divided into chunks and converted into embeddings using the LoRA-enhanced model, which are then stored in a vector database. So here, LoRA is used to improve how documents (and the user queries) are represented in the vector space to improve the retrieval rather than to directly store the knowledge in the model that generates answers.

Another hybrid approach applies LoRA after the retrieval step [44]. In the study by Su et al., such methods are called parametric retrieval-augmented generation (PRAG). In PRAG, retrieved documents are, rather than being inserted into the prompt as extra text for context, instead turned into LoRA-based parameter updates that are added to the language model during the generation step. So in other words, PRAG inserts the retrieved knowledge into the model parameters rather than using it as input context. A benefit of PRAG is the reduced need for very long prompts that may happen with normal RAG, but the early PRAG methods require one LoRA adapter per document which makes them very expensive to train and store as the number of documents grows. To address this problem, Su et al. propose so-called Poly-PRAG that uses a small shared set of LoRA adapters for the whole document collection. Then each document is represented through a weighted combination of the shared adapters [44] with the goal to make use of knowledge that is shared across many documents more efficiently. Their result shows that this design can improve both storage efficiency and performance in comparison to early PRAG. The study also brings up the potential usage of LoRA within the retrieval stage as an open and promising research direction.

Yet another hybrid design is proposed by Asai et al. (2024) called SELF-RAG [4]. This method trains a model to be able to determine when retrieval is actually useful. SELF-RAG performs retrieval when the model determines it is needed and it uses a

form of self-reflection to do this. This approach doesn't use LoRA, but it shows that parametric learning can be used to improve a retrieval-based system in this way as well.

Overall, these studies suggest that the design choices relating to methods using parametric knowledge and non-parametric knowledge do not have to be mutually exclusive, and that there are many creative ways to combine the two philosophies in order to create a better overall system.

2.3 Capacity, Scaling, and Evaluation

This section describes how the performance of parametric and non-parametric approaches of handling knowledge in language models depends on capacity, scale, and evaluation. In particular, it focuses on how parametric capacity, retrieval quality, and task difficulty interact, as well as how these aspects can be evaluated.

2.3.1 Capacity and Scaling

In this thesis, capacity refers to how much information a system can make use of when answering user prompts. For parametric approaches this depends on how many facts can be stored within the model's parameters [37]. For non-parametric approaches (retrieval approaches) this depends on how well the system can retrieve the correct pieces of information from the external database (the correct chunks) and then make use of it in the generated answer [23].

One important factor for parametric systems is the model size [37]. Roberts et al. (2020) show that larger language models are better at answering fact questions without external context than small models which in turn suggests that larger models can store more facts within their weights. Meanwhile earlier work has shown that fine-tuning can often work well in a relatively low-dimensional subspace which provides an explanation as to why PEFT methods can be so effective (since they work by only updating a smaller number of parameters than the full number of the base model's weights) [1]. In simple terms, this means that it is often sufficient to only make weight updates in a small number of directions since the base model already contains most of what it needs. This in turn highlights the importance of the choice of rank in LoRA, since it directly controls how large (and thus how flexible) the trainable adapter is [14]. A higher rank means more parameters are updated, whereas a smaller rank should in theory enforce a bottleneck on how much and how nuanced information can be internalized since fewer parameters are updated. Recent work has explored how much information can be internalized/"packed" into a LoRA adapter before the model starts to decrease in overall performance, thus tying directly into the question of where the practical capacity limits of LoRA adapters given different ranks are [32].

The LoRA rank is in practice the maximum rank of the learned update matrix. It is not the maximum rank that is used by the learned matrix, or in other words, it is not

the actual number of update directions used. It can therefore be interesting to look beyond the rank value chosen as a hyperparameter, and also consider the true usage of available update directions. Roy and Vetterli propose an *effective* rank measure [38]. After performing singular value decomposition (SVD), the singular values are normalized and thus turned into a discrete probability distribution. Then, the entropy of this distribution is computed (where a low entropy means the distribution is concentrated on a smaller number of directions, and vice versa). Finally, the entropy is passed through the exponential function, giving the effective rank of the matrix.

Other work that has investigated the singular values of trained LoRA adapters shows how one can, by only keeping the largest singular values of a weight matrix, reconstruct a smaller LoRA adapter based on the trained one [2].

In non-parametric systems (retrieval systems), other factors are particularly influential for the capacity of the system. Since knowledge in RAG systems is stored in an external database and accessed via the retriever step, the capacity of the system depends in large part on the retriever and how well it can find and return relevant pieces of information (chunks) for a given user query [23]. So in contrast to how the capacity in parametric systems scales with base model size (and for LoRA specifically with rank size), retrieval-based systems scale with the effectiveness of the retrieval stage and with how easy or difficult it is for the retriever to select the correct pieces of information from the external database and turn them into good answers during generation [23, 37].

2.3.2 Evaluation of Knowledge Recall

Knowledge recall (that is, the ability to respond correctly given a user query) can be evaluated in different ways. It can for example be done by using "cloze-style prompts" (fill-in-the-blank questions) in which one measures if the correct answer is among the most highly ranked possible outputs [31]. It can also be done by using "free generation", where one then compares the freely generated answer to a gold answer [37]. These two approaches test different aspects. Cloze-style prompts allow evaluating whether the fact "exists" in the model and can be recalled by it, whereas free generation evaluates if the model can produce a correct answer in a more realistic setting using its internalized knowledge (this is more realistic since in real-world usage, systems like these generate answers freely and are expected to provide accurate answers).

In evaluation using cloze-style prompts, the system is given a statement with the fact being masked/removed, and the system is then evaluated on how well it performs in terms of ranking the correct answer highly - for example, the proportion of times the system ranked the correct answer highest, or the proportion it ranked the correct answer in the top 5, and so on [31]. For free-generation evaluation, the model is allowed to freely generate an answer like it would in a real question-answering

context, and then evaluated on whether it generated a correct answer [37]. The evaluation can be on exact matches, which is easily interpretable and may work well in settings where the answers are short. However, this risks underestimating the system's performance since small formatting incorrectness or paraphrases lead to an incorrect mark. Thus, more lenient evaluation approaches may be preferred in some settings.

2.4 Structured Data and Similarity Measures

In this section, the concepts of structured document collections and similarity measures are discussed.

2.4.1 Structured Document Collections

The data used in this thesis take shape as structured documents. This type of data is not simply large amounts of free text - rather, it contains more structure which takes the shape of metadata such as author or publication year [24, Section 6.1]. In other words, any piece of text in the structured document collection belongs to a location given by this metadata. Retrieval over structured documents therefore sits between regular database search and unstructured retrieval over a long list of chunks, since every chunk in a structured document corpus belongs to a location given by the metadata fields. Retrieval in structured document collections is therefore both concerned with the textual aspects of a given piece of data (what the text says) as well as the structural aspects (where the text is found) [24, Section 10]. Moreover, when using retrieval on structured documents, only the relevant parts of a document are often wished to be retrieved, and in principle, a retrieval system should always retrieve the most specific part of a document answering the query [24, Section 10.2]. The granularity of the retrieved unit is thus a design choice where if it is too broad, irrelevant information may be retrieved whereas if it is too slim, it may miss relevant information. Furthermore, Manning et al. state that explicitly making use of the document structure during retrieval, such as by preferring text passages in the right location, can help the performance of the retrieval system by lowering the chances that passages containing the right textual information, but from the wrong context, are retrieved [24, Section 10.4].

2.4.2 Similarity Measures and Confusability

Structured retrieval is affected by textual similarity: documents and/or text passages in the form of chunks can be represented as vectors which are compared to the query vector using cosine similarity, after which the highest-scoring passages are retrieved [24, Section 6.3]. Therefore an incorrect text chunk may be retrieved if it is similar enough to the correct text chunk.

Similarity between documents can be measured in different ways. One way is to consider the overlap between tokens in a pair of texts. Manning et al. describe a token as an instance of a sequence of characters, or more simply put, an instance of a

word [24, Section 2.2]. When tokenizing a text, certain stop words may be omitted that do not contain much semantic value, such as "to". Measuring the overlap between tokens is then to compute how many tokens they share, or simply put, how many similar words they use. Furthermore, one may estimate similarity with Jaccard similarity [24, Section 3.3]. Jaccard similarity is defined as the intersection between two sets divided by their union, and therefore it captures the share of shared tokens (if applied to sets of tokens) but it does not capture the frequency each text uses a given token. It also treats all tokens equally.

Another way for measuring text similarity is by considering text documents as vectors in a shared vector space where the vectors contain weights for terms [24, Section 6.2-6.3]. When considering text documents as vector representations, similarity may be computed with cosine similarity which is defined as the dot product of two vectors, divided by the product of their Euclidean lengths (to compensate for the effect of vector lengths) - in other words, the dot product of the normalized versions of two vectors. In a nutshell, cosine similarity is the same thing as the cosine of the angle between the two text vectors, so a value close to 1 means the vectors are very similar. The cosine similarity can then be used for ranking during retrieval, with the text chunk vectors being closest to the query vector being retrieved.

Moreover, in this thesis, the term "confusability" refers to when two text passages, one correct and one incorrect, are similar enough to where retrieval has a harder time ranking the correct passage above the incorrect passage. Token overlap and Jaccard similarity measure this at the vocabulary level, and cosine similarity measures it at the vector space level. Related to the notion of confusability is so-called "hard-negatives", which in this thesis means text passages that, for a given query, are incorrect but that are highly confusable with the correct text passage. Xiong et al. discuss how incorrect passages that are drawn from the retrieved passages are most interesting, since these are the passages the system must be good at distinguishing from the correct passage [48].

3
Method

3.1 Method Overview

The study aimed to answer the research questions by testing four different systems: (1) a system using LoRA-based fine-tuning on a base model for knowledge internalization, (2) a RAG baseline system, (3) LoRA-adapted retrieval in a RAG system, and (4) a LoRA-trained context-routing system with RAG.

Overall, the experimental pipeline began with the generation of a set of synthetic and structured datasets, followed by system-specific data processing as needed. The original datasets were constructed as canonical fact sources consisting of structured text similar to company information. From these datasets, system-specific material such as question–answer pairs and vector embeddings was created (depending on the system).

Each system was tested under controlled experimental conditions with both shared and system-specific variables, with the aim to allow for as fair comparisons as possible while acknowledging system-specific differences. Across all systems, experiments were performed on the shared datasets with systematic variation in dataset size (N) and confusability (S). Moreover, for the systems using LoRA, the LoRA-specific settings were shared across systems. In these systems, rank was varied systematically, and the same number of epochs was used to ensure consistency. This means the experiments used a fixed number of epochs rather than a fixed number of optimization steps, to ensure equal fact exposure across datasets. Therefore larger datasets received more update steps than smaller ones.

All experiments were performed on rented GPU instances via Vast.ai, with runs launched and managed through Mac terminal environments. Outputs from each run were stored on GitHub. Separate GitHub branches were maintained for each system, as well as for the data generator, to ensure modularity.

To facilitate the programming of each system as well as the results analysis and plotting, AI agents Codex by OpenAI and Claude Code by Anthropic were used [27, 3].

A list of the experimental runs performed can be found in Appendix A.2.

3.2 Data

This section describes the data used in the experiments.

3.2.1 Explanation of the shared synthetic data

The datasets used for the experiments are entirely synthetic. The shared datasets were designed to be similar in structure to typical documents handled by companies such as annual reports, earnings call documents, and so on. In completely different document contexts, such as novels or scientific papers, similar structure may be seen. Thus, this document structure was designed to mimic documents in general. By using synthetic data, the properties of the data can easily be manipulated - such as size and content. It also removes any potential data privacy issues. Although some realism is lost through this approach in terms of the general variability that can be expected in real company documents, this setup allows for a reasonable balance between control and realism.

Investor Presentation

Yorvale Systems, 2021

Sustainability Highlights

Supported by a targeted initiative, the company reduced a sustainability metric by 9.0% during the period. Notably: An operating lever helped the company lift a key operational KPI by 6.5% during the period.

The company presented the demand business view as mixed across regions, particularly in OEM partners. A targeted initiative helped the company increase a sustainability metric by 6.3%.

Driven by service contract renewals, its reported revenue climbed by 9.4% during the period. In addition: Driven by renewable electricity sourcing, the company reported a renewable electricity share increase of 11.6% during the period.

A key investor message was this: The company reported a solidity decline of 1.6 percentage points, driven by inventory build-up. Another presentation takeaway was that the company cited service-led expansion as a major growth lever, with notable support from a core segment.

The company cited automation upgrades as a major expansion lever, with strong contribution from a core segment. Its expansion was driven by a key business driver, with the strongest activity in a core segment.

Figure 3.1: Illustrative example of a synthetic document.

It may be beneficial to think of the data in terms of *contexts*. A context is a specific "place" in the dataset, in a sense. For example, the text in Figure 3.1 belongs to

the context *YorvaleSystems/2021/InvestorPresentation/SustainabilityHighlights*. A context always has 5 paragraphs, and each paragraph contains 2 facts. This is by design to ensure consistency and to enable control over what aspects of the data change when increasing the number of contexts. The data generation is designed to ensure that no facts ever contradict each other, and that the text is semantically sound.

Two major, "global", variables were varied in experiments for all systems. The first one is **N** - the number of contexts. Increasing N means there are more "places" where information can be located in the dataset, and since a context always has 5 paragraphs and 2 facts per paragraph, it also means the number of facts increases proportionally.

The second variable is **S** - the measure indicating confusability. This is slightly more complex than N. The overall purpose of S is to enable changing how confusable the data is, or equivalently, how easy it is to distinguish the correct paragraph from possible alternatives. In contrast to N, which may take any positive integer value, S is designed to only be set to "low" or "high". S changes two aspects of how the dataset is constructed: *surface similarity,* and *distractor confusability*. Surface similarity means how lexically similar paragraphs are to each other - in other words, paragraphs are more similar to each other on the surface, while the actual semantic content may remain different. A higher S thus means more shared vocabulary and more similar sentence structure, but not necessarily more shared topics. Distractor confusability on the other hand means that the actual contexts become more similar, meaning a high distractor confusability leads to paragraphs more often being similar in terms of topics and themes. For example, paragraphs may more often belong to the same company and year, or they may more often talk about similar metrics. Also, a higher S reduces the explicitness of some words that would be particularly good for discriminating between paragraphs - for example a query may ask about a specific customer segment, but the paragraph containing the fact may not explicitly mention the customer segment by name. Taken together, a higher S ought to make retrieval more difficult in the RAG-systems, since the embeddings become more confusable.

3.2.2 The eight shared datasets

For the main experimental comparisons between the four systems, eight shared datasets were generated in the style described above. These datasets correspond to four values of $N \in \{12, 30, 90, 300\}$ and two confusability settings, $S \in \{\text{low}, \text{high}\}$. Data generation for these eight datasets was performed by sampling from the same 30 companies, 8 years, 6 document types, and 8 section types.

N	Contexts	Paragraphs	Facts	Unique facts	CY coverage	Indexed chunks	Indexed facts
12	12	60	120	120	5.0%	60	120
30	30	150	300	300	12.5%	150	300
90	90	450	900	900	37.5%	450	900
300	300	1500	3000	2693/2710	100.0%	1287/1291	2574/2582

Table 3.1: Effect of dataset size N on scale and structure of the shared datasets.

Table 3.1 provides an overview of the dataset size regimes used in the shared data. "Unique facts" are less than "Facts" for the N300 datasets - 2693 for the low S setting and 2710 for the high S setting. This is because at N300, some facts started to be repeated in new contexts. This is connected to the "CY coverage", which means how many of the available Company-Year combinations were used by the dataset (30 companies times 8 years = 240). At N300, the 240 combinations were saturated and started to be reused across different document types and section types. Thus, the same fact for a given company and year may be stated in two different documents for that company and year. Moreover, this effect is seen in "Indexed chunks" and "Indexed facts". Since paragraph chunking was applied, the number of indexed chunks would normally be equal to the number of paragraphs. But due to the fact saturation at N300, chunks that repeat the same facts were pruned off in order for the N300 experiments in retrieval to not include a confound. For example, if the retriever retrieves a chunk including the correct fact, but it was not marked as the gold chunk, then the retrieval would be marked as incorrect falsely. Similarly, if all chunks containing the correct fact would be considered gold chunks, then retrieval for those queries would be easier for N300 than the other datasets, and comparison wouldn't be fair then either. Thus, this pruning was done for fairness reasons. An effect of this is that the number of indexed facts is less than the number of unique facts at N300. This is because each chunk contains two facts, and thus pruning a chunk inevitably removed some unique facts that didn't need to be removed.

N	Hard-neg cosine	Hard-neg Jaccard	Hard-neg token overlap	Queried name stated
12	0.2406 → 0.3114	0.2120 → 0.2642	0.3365 → 0.4087	0.859 → 0.364
30	0.2576 → 0.3161	0.2149 → 0.2924	0.3432 → 0.4378	0.868 → 0.311
90	0.2950 → 0.3502	0.2375 → 0.3060	0.3727 → 0.4600	0.865 → 0.403
300	0.3062 → 0.3488	0.2566 → 0.2964	0.3956 → 0.4503	0.887 → 0.388

Table 3.2: Effect of the confusability setting S on similarity and paragraph explicitness of the shared datasets.

As can be seen in Table 3.2, for fixed N, increasing S has effects on the metrics in the columns. Here, "hard-neg" means hard negatives: for any given context, there is a set of other contexts that are intentionally similar to it, meaning they are meant to be harder to distinguish from that context in terms of semantics. "Hard-neg cosine" in Table 3.2 means the average cosine similarity between the text in a context (all 5 paragraphs) and its associated hard-negatives, where a higher value means they are more similar. "Hard-neg Jaccard" and "Token overlap" mean the same thing,

both being calculated using contexts with their hard-negatives, but reporting the Jaccard similarity and the token overlap respectively. "Queried name stated" means that, among the test queries that ask about a specific customer segment or region, how often the paragraph with the answer also explicitly mentions that customer segment or region by name. Overall, Table 3.2 shows that S has a tangible effect on confusability by increasing similarity between contexts while also making the paragraphs less explicit about which segment or region the facts concern.

3.3 System-Specific Methods

Below are system-specific methodologies described. Exhaustive tables for the system-specific settings used can be found in Appendix A, tables A.1, A.2, A.3, and A.4. Note that the system settings described were chosen based on recommendations found in external sources for what are typically sound choices. However, no exhaustive research phase was performed to find the optimal settings for each respective system, since that would be outside the scope of the study.

3.3.1 System 1

System 1 is designed to investigate LoRA's ability to internalize new knowledge within its parameters - in other words, the system is intended to work as a memory for trained facts, using LoRA. In simple terms, System 1 is focused on fine-tuning a small adapter to learn the new knowledge, and the adapter is added on top of the LLM's existing model weights. System 1 is primarily trained on question-answer pairs derived from the eight shared datasets, and then evaluated using new formulations of those same facts. An example is shown in Figure 3.2. For each fact, there are two train QA-pairs, one validation QA-pair, and one test QA-pair, meaning for each training epoch the model is exposed to an individual fact twice.

Train data

```
Question:  What was the market outlook for Xandrel Equipment in 2024
in Europe?
Answer:  moderating after a strong prior year

Question:  How was the market outlook for Xandrel Equipment in 2024 in
Europe?
Answer:  moderating after a strong prior year
```

Validation data

```
Question:  What outlook did Xandrel Equipment report for Europe in 2024?
Answer:  moderating after a strong prior year
```

Test data

```
Question:  What market outlook was reported for Xandrel Equipment in
2024 in Europe?
Answer:  moderating after a strong prior year
```

Figure 3.2: Example of train, validation, and test question-answer templates for the same underlying fact in System 1, derived from one of the eight shared (semantic) datasets.

The system is not evaluated on new information - rather, it is evaluated on whether it is able to correctly answer questions in new wordings concerning the same facts that it has seen during training. During evaluation, the model deterministically generates a text answer freely, which is compared to the gold answer.

System 1 was also used on a separate set of data, called **"stress data"**. This data lacks the semantic realism that the shared datasets have, and instead replaces companies, attributes, and values with IDs. The purpose of this stress data was to examine how System 1's performance is affected when it no longer can benefit from learning the semantic domain of companies and company-relevant topics. It therefore functions as a "stress test" on LoRA's ability as a tool for pure fact memorization. The stress data follows the same sizes as the shared datasets, taking values of N of 12, 30, 90, and 300 which corresponds to the number of contexts. For these datasets, the corresponding number of facts is 120, 300, 900, and 3000. Moreover, the answers for a given dataset begin with a dataset-specific prefix such as "4K" for N12, followed by two (consonant) letters and two numbers which then form the unique answer for a given context and attribute. The reason for omitting vowels is to avoid accidentally forming words.

Train data

```
Question:  What value is assigned to attribute ATR_S3J in context
CMP_SQG / IND_DVQ / DOC_3M / SEC_X9?
Answer:  4KHT98

Question:  In context CMP_SQG / IND_DVQ / DOC_3M / SEC_X9,
what is the value for attribute ATR_S3J?
Answer:  4KHT98
```

Validation data

```
Question:  For context CMP_SQG / IND_DVQ / DOC_3M / SEC_X9,
which value corresponds to attribute ATR_S3J?
Answer:  4KHT98
```

Test data

```
Question:  Within context CMP_SQG / IND_DVQ / DOC_3M / SEC_X9,
what value is recorded for attribute ATR_S3J?
Answer:  4KHT98
```

Figure 3.3: Example of train, validation, and test question-answer templates for a symbolic stress fact in System 1.

Across experiments for the eight datasets as well as for the stress data, rank was varied and took the values 1, 2, 4, 8, and 16. In evaluation, each output from System 1 was assessed on whether it was correct or not. Due to the infeasibility of manually scoring every model output during the evaluation step, a script was used for scoring. Specifically, the script scores a prediction as correct if the gold answer is contained within the output string - meaning the model is allowed to answer in different formats or ramble, but as long as the gold answer exists within that answer, it is correct. If the model output includes an abstention "I don't know", then that counts as an incorrect answer regardless if the output also contains the gold answer. GIP is not a perfect performance metric for a language model. For example, it doesn't account for the possibility of semantic confusion - there could be cases where, for instance, the gold answer is preceded by a "not" or a "no", making the model's semantic output incorrect although it would still count as a GIP success - upon visual inspection, this appeared only a few anecdotal times in a way that didn't materially affect the total GIP of a run. Another drawback is that GIP doesn't assess the total answer quality - for example, the answer may contain other misleading or incorrect statements, or the model may start to give multiple answers to the question in the hope of one being correct (this risk is mitigated through the way training is set up). Connecting to the theory section regarding evaluation of knowledge recall, GIP can be understood as a variant of exact-match evaluation. Overall, GIP should be interpreted as a reasonable proxy for knowledge recall in the tested systems rather than a perfect measure of answer generation quality.

3.3.2 System 2

System 2 is the simplest system in the project, acting as the Retrieval-Augmented Generation (RAG) baseline. Instead of internalizing knowledge in model parameters during a training phase, the system retrieves text chunks from a vector database to use as context for the answer generation. Further details on RAG are described in the theory section of this report.

The dataset was first turned into a collection of small text chunks, by letting each paragraph in the document collection be one chunk. Each chunk contains two facts. For example, a paragraph turned into a chunk might look like:

> "In 2022, the Xandrel Equipment reported revenue growth of 7.6%, driven by strong demand in European markets. Operating margins also improved due to cost reductions."

From this data, questions were created that correspond to one specific chunk. For example:

> **Question:** What was the revenue growth for Xandrel Equipment in 2022?
> **Answer:** 7.6%

As stated, in contrast to System 1, System 2 does not try to answer from memory. It first tries to find the chunk where the answer is, and then the predicted answer is generated from the retrieved candidate chunks. After retrieval, it was checked whether the gold chunk was among the returned chunks. The goal for retrieval is to rank the gold chunk as highly as possible. Retrieval performance is measured using three metrics, which in this thesis are defined as follows - first, **Hit@1**, the share of test queries where the gold chunk was retrieved at the number one ranking; second, **Recall@5**, the share of test queries where the gold chunk was ranked within the top five retrieved chunks; third, **Mean Reciprocal Rank (MRR)**, a weighted metric favoring gold chunks being ranked highly. These were inspired by standard retrieval evaluation metrics, tweaked to suit this experimental setting [13, 12]. For example, if the correct chunk is ranked first, retrieval succeeded perfectly. If it is ranked third, the system still finds it but less precisely. If it does not appear in the top 5, the retrieval fails. Generation, meaning the output given by the system as the answer for the user query, was evaluated with the same metric as System 1 with gold-in-prediction (GIP).

Example retrieval setup

`Question: What was the revenue growth in 2022?`

`Top 5 retrieved chunks:`

- Chunk 1: (GOLD CHUNK) "...revenue growth of 7.6%..."
- Chunk 2: "...operating margin improved by 2%..."
- Chunk 3: "...revenue growth of 5.2%..."
- Chunk 4: "...market outlook remains strong..."
- Chunk 5: "...risk factors include supply chain issues..."

Figure 3.4: Example of how System 2 retrieves chunks for a given query.

Importantly, nothing was trained in this system. The embedding model that embeds chunks and queries into vector representations is completely frozen. This makes System 2 a clean baseline for comparisons against System 3 and System 4, which use LoRA in two distinct ways to attempt to improve the baseline System 2 performance. In other words, System 2 shows how well retrieval works without any task-specific learning and provides a reference point for the more advanced systems that follow.

3.3.3 System 3

System 3 is, in principle, an extension of System 2 in the sense that it uses the same base RAG system and is evaluated using the same data. The difference lies in that it fine-tunes the embedding model with LoRA, with the purpose to evaluate whether this can improve retrieval performance and, in the end, the ability of the generator to output a correct answer to user queries.

System 3 was trained using contrastive learning and a cross-entropy training loss, with query-chunk pairs derived from the eight shared datasets (rather than QA-pairs like System 1). More specifically, during training, it was exposed to a train query and five chunks: the associated gold chunk, and four uniformly sampled incorrect chunks. As a result of the fact that the model was exposed to randomly selected chunks rather than meticulously selected hard negatives, the training objective mainly encourages the model to separate the gold chunk from generally unrelated chunks - this is slightly different from the inference-time evaluation task that requires the retriever to distinguish the gold chunk among more difficult negative chunks. Nevertheless, when given the query-chunk pair and the incorrect chunks, the embedding model embeds the query and the chunks into vectors which are L2-normalized, and the cosine similarity is then computed between the query and the five chunks. Then, temperature is applied, and the values are passed through the softmax function to be converted into probabilities (i.e., "the probability that a given chunk is the gold chunk for the query"). The loss function is then the negative log of the gold chunk's assigned probability, which the optimizer minimizes through gradient descent. Note that System 3 was trained on 20% of the total data, and is thus trained on other

facts than it was evaluated on but using similar query wordings in training and testing.

After training was completed, the modified LoRA-embedder was used to build the FAISS chunk index and to embed test queries. System 3 was evaluated, both in terms of retrieval and in terms of generation, using the same metrics as System 2.

3.3.4 System 4

System 4, similarly to System 3, tries to improve the baseline RAG system (System 2) using LoRA, but in a very different way. It extends the retrieval setup in System 2 by adding a filtering step before retrieval. Specifically, the system adds a separate neural network that is trained with LoRA to predict the metadata associated with the gold chunk given a user query. This is like applying filters before searching. Rather than including all chunks from the dataset during retrieval, the system first predicts the context it believes the gold answer exists in, and then the system filters on that context and performs retrieval on that subset of all of the chunks.

Every chunk in the eight shared datasets belongs to a context Company/Year/Report/Section, and in each dataset there's five chunks belonging to any existing such context. The router model works as a multiclass classifier, that reads the query and predicts a context label. The router outputs probabilities over all existing context labels, and the system selects the most likely one. This predicted context is then used as a filter. Only chunks that belong to this context are kept, while all others are ignored. The model is trained as a multiclass cross-entropy optimization problem, where the training queries are passed through the model that then outputs a logits vector turned into a probability vector through the softmax function, which represents the predicted probability for each label, after which the cross-entropy loss is computed and model weights are updated through standard backpropagation.

After filtering, retrieval is performed in the same way as in System 2.

To control how strong the filtering is, **four routing levels were used**: Company, Company + Year, Company + Year + DocumentType, Company + Year + DocumentType + Section. These levels determine how large the filtered subset becomes. During training, the selected routing determines what the gold label is (meaning the training task becomes more "specific" as routing level becomes more granular). Coarser levels mean retrieval includes more chunks, while more granular levels produce smaller subsets of chunks. Before any rank sweep was performed, experiments were conducted across these four routing levels at rank 16, to provide an indication of the performance ceiling of each routing level and to enable a feasible analysis of the performance of each routing level. A rank sweep was only performed for the most promising routing level.

For evaluation, the same metrics for retrieval and generation respectively were used

as for System 2 and System 3.

Example

`Query: What was the net sales for Company X in 2022?`

Step 1: Filtering

`Keep only: Company_2022`

Step 2: Retrieval

- Chunk 1: (GOLD CHUNK) "...net sales were 320B..."
- Chunk 2: "...operating margin improved..."
- Chunk 3: "...cost reductions..."

Step 3: Generation

`Answer: 320B`

Figure 3.5: System 4 pipeline: filtering, retrieval, and generation. Here, Company+Year routing level is used.

3.3.5 Contrasting settings

The following table highlights the settings that are shared across at least two systems, and where the setting differs. The full settings for each system along with motivation for setting choices can be found in Appendix A, tables A.1, A.2, A.3, and A.4. Note that the systems are designed to investigate different possible roles for LoRA in QA-systems. Comparisons across systems should be read with that in mind - they do not claim to solve the same learning problems.

Table 3.3: Contrasting settings across the four systems.

Component	System 1	System 2	System 3	System 4	Explanation
LoRA-trained component	Fine-tuned generator (no retrieval, parametric memory only)	None, standard RAG pipeline	Fine-tuned embedding model in a standard RAG pipeline	Fine-tuned classification router added on top of a standard RAG pipeline	Core design choices for evaluating different approaches
Training target	Output the correct answer text (language model fine-tuning)	-	Place query embedding close to the relevant chunk (contrastive learning)	Identify correct route (classification task)	Follows from the distinct design choices
Training data	QA pairs derived from facts in the shared synthetic datasets and the System 1-specific stress data	-	Query–gold chunk pairs and uniformly sampled negative chunks from the shared synthetic datasets	Query–route label pairs derived from chunk metadata from the shared synthetic datasets	Each LoRA system is trained with supervised learning in distinct ways, which follows from each system's role and design
Optimization steps	∼350-7000	-	∼175-3500	∼175-3500	System 1 is exposed to the same fact in two different query formulations, a confound to be aware of when comparing results across systems
Evaluation	Cross-system evaluation queries, which are new question formulations of the same facts seen during training (facts seen, new formulations)	Cross-system evaluation queries	Cross-system evaluation queries (facts not seen during training in the query–gold chunk pairs, but facts sometimes present as negative chunks in contrastive learning)	Cross-system evaluation queries (facts not seen during training, but route labels are seen)	Follows the design choices for the systems. The systems should thus not be interpreted as solving identical generalization tasks, although they are all evaluated on the exact same questions
Retrieval strategy	No retrieval	Global retrieval	Global retrieval with the adapted embeddings	Retrieval filtered based on routing	Only System 1 is not a retrieval system; the others vary in how chunks are retrieved
LoRA-adapted model	Qwen2.5-0.5B	-	Qwen/Qwen3-Embedding-0.6B	Qwen/Qwen2.5-0.5B with sequence-classification head	Different choices made due to the differences in the task with the main consideration to use models of similar sizes
LoRA ranks	$r \in \{1, 2, 4, 8, 16\}$	-	$r \in \{1, 2, 4, 8, 16\}$	$r \in \{1, 2, 4, 8, 16\}$ for Company+Year routing; $r = 16$ for remaining routing levels	Due to constraints in scope, a rank sweep was only conducted on the most promising routing level for System 4. Since the routing levels added a further "dimension" to the study, focusing on one routing level enabled a more intuitive comparison between systems
Model for answer generation	Qwen/Qwen2.5-0.5B	Qwen/Qwen2.5-0.5B-Instruct	Qwen/Qwen2.5-0.5B-Instruct	Qwen/Qwen2.5-0.5B-Instruct	Non-instruct model chosen to act as maximally pure baseline for fine-tuning in System 1; instruct versions used in remaining systems since the generator is not fine-tuned there

3.4 Analysis approaches

Below are the metrics and approaches used for analyzing the results described. Beyond what's described below, the results were also analyzed using line charts and bar charts to describe relationships found between things like LoRA rank, dataset size, performance metrics, and so on.

3.4.1 Correlation analysis

To analyze the correlation between e.g., loss and performance metrics, scatter plots were used for visualization. Moreover, the following were used: Pearson correlation coefficient r, the associated two-sided Pearson p-value, and Mutual Information. The first two are metrics for the linear correlation between two variables, and the third is a metric that captures nonlinear relationships.

Pearson r value is a measure for the linear relationship between two datasets [46]. It takes a value between -1 and 1, where -1 means there's an exact negative linear relationship and 1 that there's an exact positive linear relationship. When analyzing whether learning the optimization task during training leads to a better end result, the Pearson r is expected to be negative and close to -1, since decreasing the training loss ought to improve system performance. Moreover, the associated p value provides an indication as to the probability that a system with no correlation between the variables produce data points that give an r value at least as extreme as the computed r value. In other words, the Pearson r value indicates the magnitude and direction of the observed correlation, and the p value indicates the confidence we have that this correlation is real and not random.

The Mutual Information (MI) metric measures statistical dependence more generally and is able to capture nonlinear relationships between variables [39, 20]. The MI metric takes a positive value, and doesn't give any indication on the direction of the relationship. It uses a k-nearest neighbors approach, for each data point it looks at its neighbors and checks whether closeness in one variable usually implies closeness in the other variable. If it does, then the MI increases because then knowing the value of one variable tells us something about the other. Simply put, MI complements Pearson coefficient in that it provides another way of estimating relationship between, for example, loss and performance, that may capture more complex patterns, but it doesn't allow us to see the direction of the observed relationship nor the confidence we may have in it.

3.4.2 LoRA adapter analysis

LoRA rank is, as described above, only a hyperparameter that dictates the maximum dimensionality of the adapter matrices. During training, the LoRA updates applied to the target modules may use fewer such update directions. Therefore, the effective rank metric as described earlier was applied to determine the approximate number of meaningful update directions used [38]. Moreover, to complement this analysis,

singular value decomposition was performed where the normalized singular values were inspected in order to see how concentrated the updates are in few or multiple directions, as described previously [2].

3.4.3 Embedding space analysis

In order to analyze the embedding space - especially relevant for System 3 that actively manipulates the embedding model - metrics such as cosine similarity are fundamental. Even though it would be preferred to visualize the embedding space to see clusters appear clearly, it is not possible even with for example principal component analysis due to the very high dimensionality of the embedded vectors. Instead, by calculating pair-wise cosine similarities, distributions were created that show how concentrated the chunk embeddings are for different embedding models (for example, the baseline model compared to the LoRA updated one). Moreover, average cosine similarity between chunks belonging to the same group or across different groups was also calculated. For example, by subtracting the average cosine similarity between pairs of chunks belonging to the same company from the average cosine similarity between pairs of chunks belonging to different companies, we get an idea of how the LoRA updates of the embedding model reshape the embedding space in order to separate company-chunks.

4
Results

This chapter presents the experimental results for Systems 1-4. Analysis and interpretation of the results are left to the Discussion section.

The results aim to answer four questions:

- How well can LoRA internalize structured knowledge directly in model weights?
- How well can LoRA improve retrieval by adapting the embedding model used for document retrieval?
- How well can LoRA improve retrieval by guiding the system toward the correct context before performing retrieval?
- How do LoRA rank requirements change as the number of contexts grows and as contexts become more confusable, across these different applications?

To answer these questions, each system is mainly evaluated using a consistent set of metrics:

- **GIP**: performance metric for the final generated answers to the test queries (System 1-4)
- **Hit@1, Recall@5, MRR**: performance metric for retrieval (System 2-4)
- **Training loss**: measures optimization behavior during training

These metrics are reported across dataset sizes N in 12, 30, 90 and 300 and confusability settings S in low and high.

The structure of the result section is as follows:

- **System 1**: focus on generation performance, split between a section for the shared *semantic data* and the System 1-specific *stress data*
- **System 2**: focus on retrieval performance and generation performance
- **System 3**: focus on retrieval performance, generation performance, and embedding geometry changes
- **System 4**: focus on retrieval performance, generation performance, and effects of routing depth and breadth
- **System comparisons**: highlight differences in performance, training behavior, and differing retrieval-generation effects

It is important to note that this section focuses on reporting the empirical patterns. The underlying reasons for these patterns, as well as their broader implications are discussed in detail in the Discussion chapter. Readers primarily interested in interpretation may therefore proceed directly to that chapter after reviewing the main figures.

Table 4.1 summarizes the main empirical findings. More detailed results are then presented in the following subsections.

Table 4.1: Overview of the results observed in the experiments.

System	Benefits observed	Limitations observed	Rank patterns
System 1	Largest observed question-answering performance in the tested setting. Learns format, and then answer pool, already at low rank – the same goes for seen-formulation queries.	Needs retraining whenever new knowledge is added. More sensitive to capacity saturation on non-semantic data.	Very sensitive to rank. QA-performance increases gradually with rank on semantic data, while there is a sharper rank threshold for non-semantic data.
System 2	Acts as the RAG reference baseline in the study, to enable comparison with the LoRA-improved versions of the same system – System 3 and System 4.	Performance in retrieval, and generation, decreases as dataset size and confusability increases.	No rank dependence.
System 3	Improves retrieval modestly but robustly in the larger dataset regime, especially improving Recall@5 performance. Generation performance improvements follow retrieval performance improvements in a predictable way. Does not necessarily need retraining as new data is added.	Less dramatic improvement in QA-performance than seen in the parametric memory experiments. The training setup tested improves Recall@5 more extensively than Hit@1.	The main improvement appears already at rank 1. Increasing rank further gives gradual improvements.
System 4	Shows large improvements in retrieval performance, with large improvements showing at medium-coarse filtering levels. Only needs retraining when data belonging to new contexts are added.	Routing failure leads to unrecoverable retrieval failure. The strong retrieval improvements do not always lead to equally strong generation improvements in all data settings.	Low rank suffices to receive strong retrieval gains at relatively coarse filtering levels that are easy for classification.

4.1 System 1

The following section reports the results from the System 1 experiments. The results are split between the semantic data experiments and the stress data experiments.

4.1.1 Semantic data

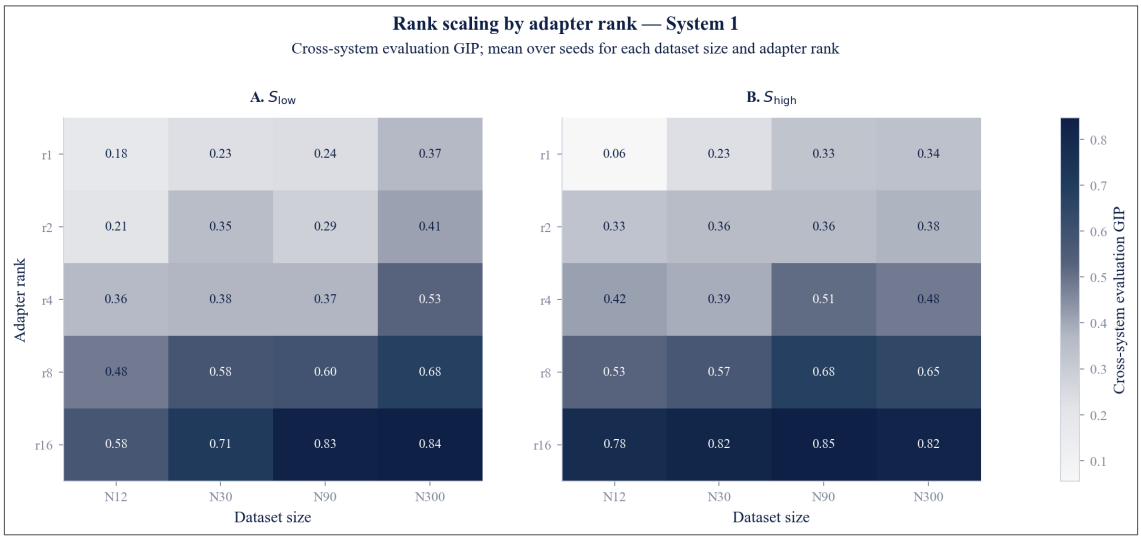

Figure 4.1: Heatmaps for System 1 rank scaling in the semantic setting.

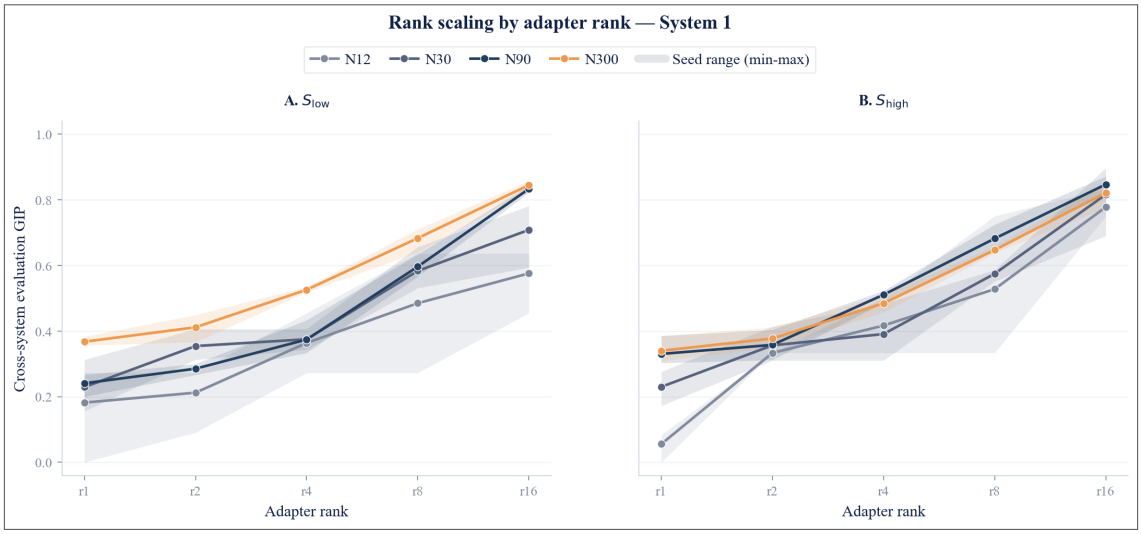

Figure 4.2: System 1 rank scaling in the semantic setting.

Figure 4.1 and Figure 4.2 show the effects for the low-confusability setting "low S" on the left as well as the high-confusability setting "high S" on the right. The rank gains are smooth and incremental when moving from rank 1 to rank 16. Increasing dataset size generally has a positive effect on GIP performance. This effect appears stronger for the low S setting than the high S setting.

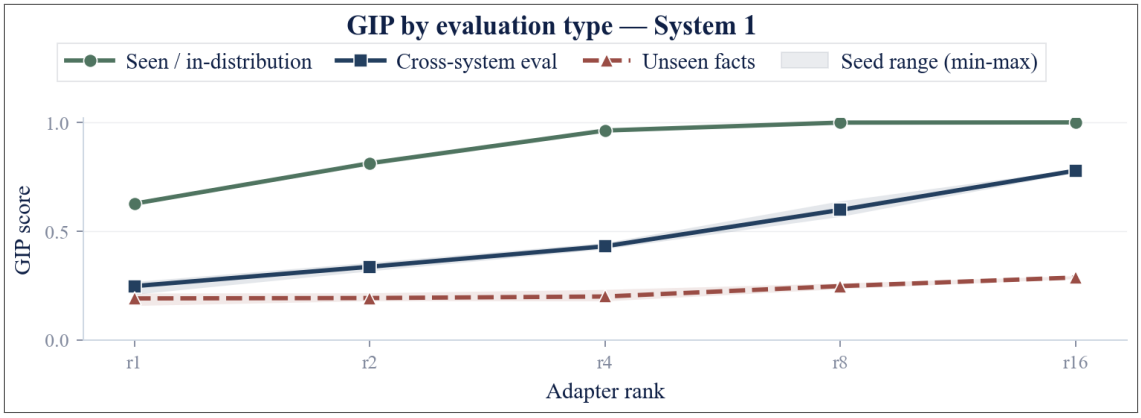

Figure 4.3: Performance by evaluation type pooled over dataset sizes for System 1 in the semantic setting.

Figure 4.3 dives deeper into how performance improves for the semantic data as rank increases, averaged across dataset sizes and S-settings. Seen distribution shows the performance when being tested on the exact same queries the system was trained on (the same formulation templates). Cross-system eval is the main performance result, showing how well the system performs on new wordings for the same facts it saw during training. Unseen facts show the performance on entirely new facts. The overall pattern is that for the seen distribution, the performance saturates at about rank 4. For the cross-system eval, performance keeps increasing and reaches closer to the performance of seen distribution at rank 16. Moreover, the performance on unseen facts improves slightly as ranks increase.

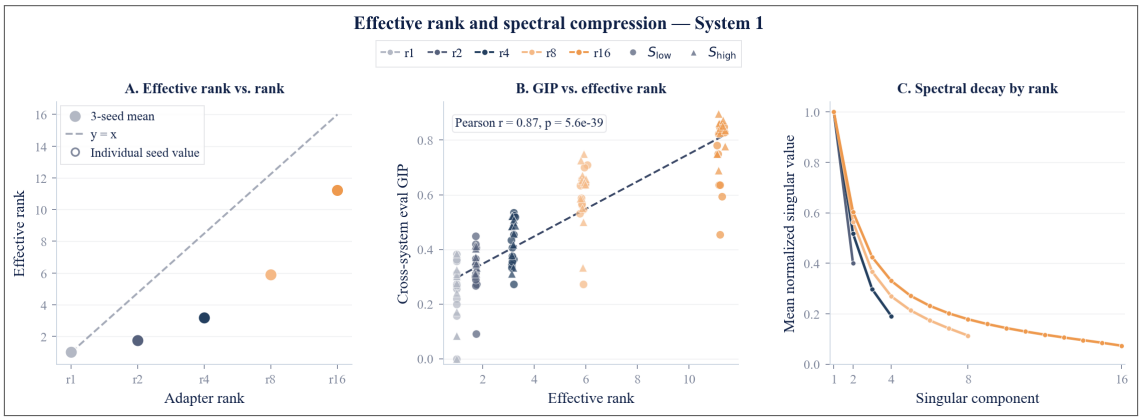

Figure 4.4: Effective rank and spectral compression for System 1 in the semantic setting.

Figure 4.4 looks deeper into the trained LoRA parameters. For each of the weight matrices for the projection modules where LoRA is applied, the approach by Roy & Vetterli described in the theory-chapter is used to determine the effective rank - in simple terms, how many update directions the model uses [38]. Panel A shows that the effective rank increases with rank, however not linearly; for example, rank 4 has effective rank circa 3.2, and rank 16 circa 11. Moreover, Panel B shows a clear correlation between the effective rank and the system performance, although there's a large spread in performance that is not explained by effective rank. Panel C builds

on the method described in [2]. On the x-axis, singular value 1 means the largest singular value, singular value 2 the second largest, and so on. The y-axis shows the value for a given singular value divided by singular value 1. This illustrates the distribution of the singular values - the steeper the curve, the less the LoRA weights use the additional directions available. The graphs show a substantial drop-off in importance of additional dimensions as rank increases, although no direction is entirely omitted.

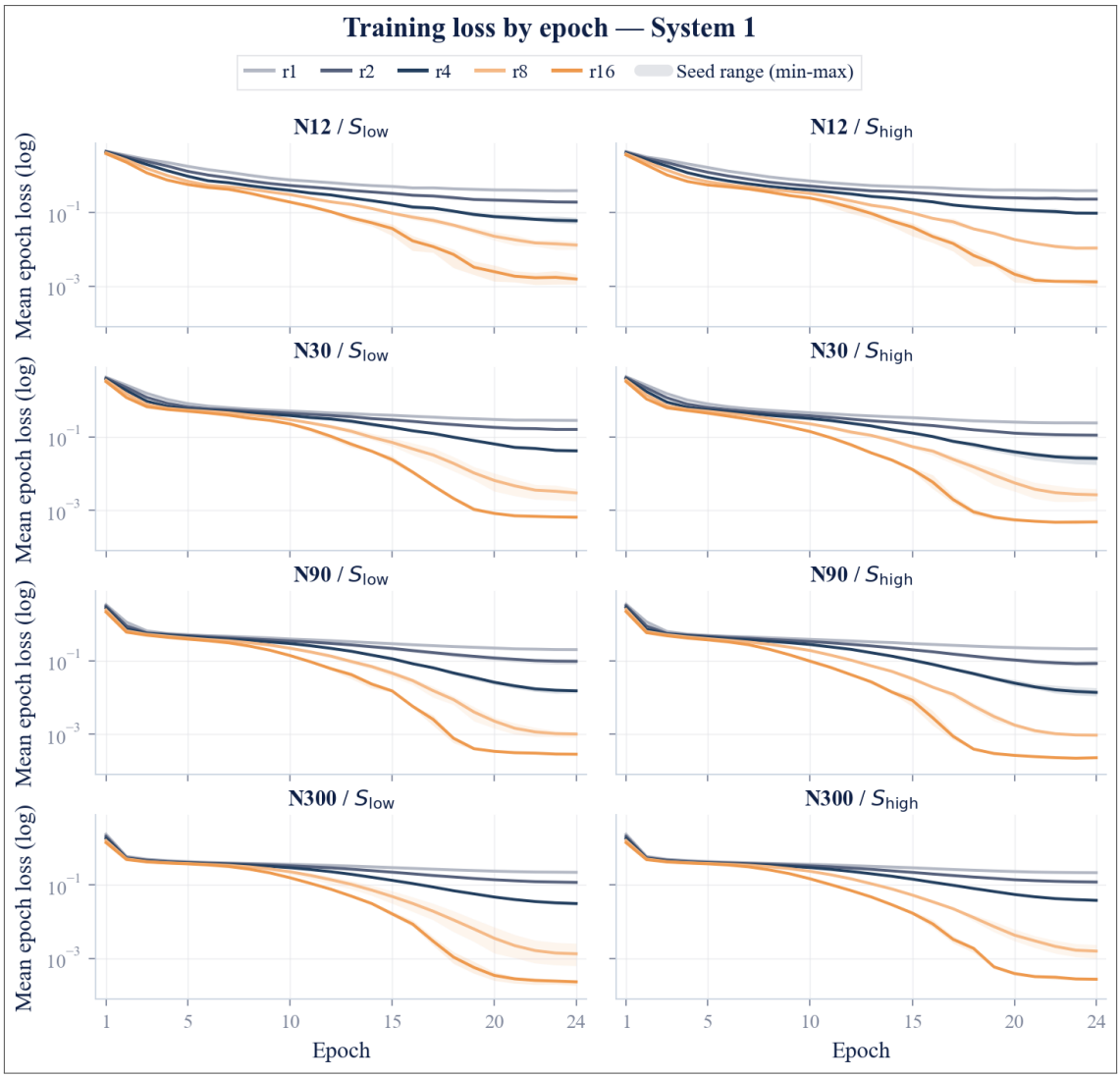

Figure 4.5: Training loss for System 1 on semantic datasets across LoRA ranks.

Figure 4.5 shows the training loss for System 1 on the semantic datasets across different LoRA ranks. The loss decreases consistently across epochs for all different sizes in a stable manner. A clear pattern is that higher ranks converge to lower training loss, with r16 achieving the lowest loss and r1 the highest. This behaviour is consistent across all dataset sizes and both confusability settings, and the gap between ranks becomes more pronounced as training progresses.

4.1.2 Stress data

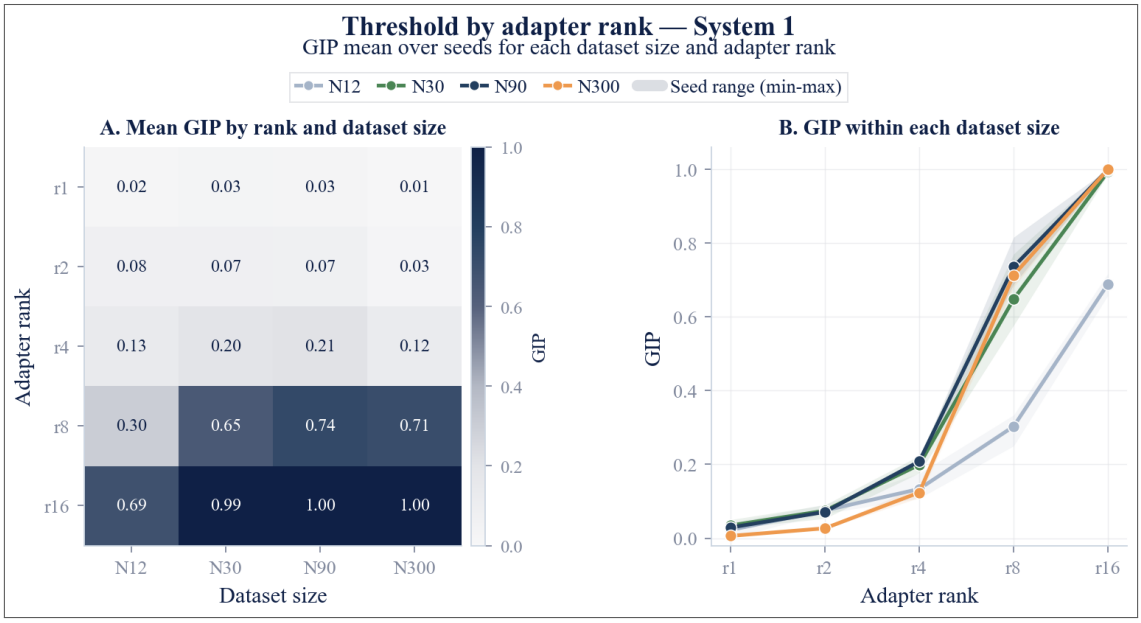

Figure 4.6: Rank scaling with observed threshold for System 1 in the stress setting.

Figure 4.6 shows the overall results of increasing rank across the different dataset sizes for the stress data. The figure shows a drastic improvement in performance when moving from rank 4 to rank 8. It also shows signs of a saturating effect for lower ranks - performance drops for rank 1 through rank 4 for the largest dataset, an effect not present for rank 8 and rank 16. Rather, especially for rank 16, increasing dataset size shows improved performance. It is also quite clear that for the smallest dataset size, N12, the system struggles to reach the same performance levels as for the larger sizes.

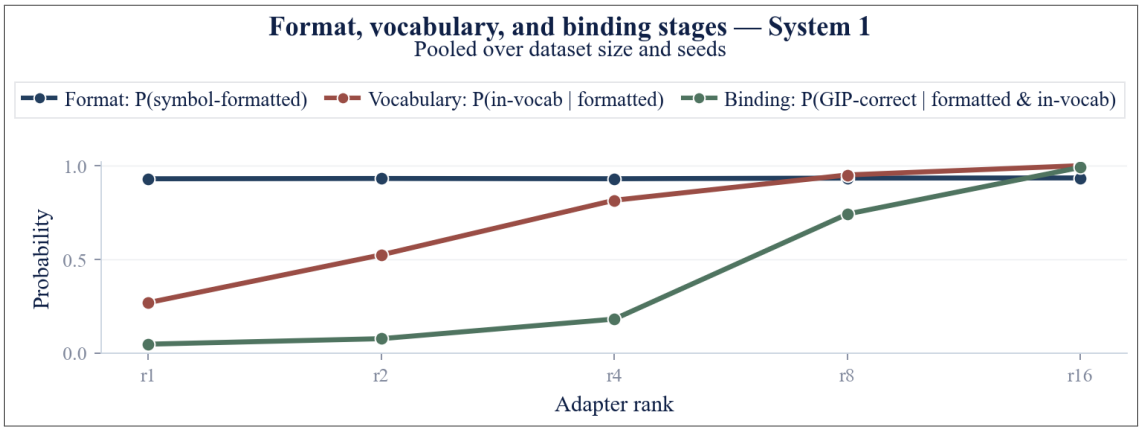

Figure 4.7: Learning stages pooled over dataset sizes for System 1 in the stress setting.

Figure 4.7 describes what parts of the task that System 1 learns as rank increases, averaged across dataset sizes. The blue line means that the system produced an

output of the expected format, such as "4KHT98" or "4KDDZ6". An output like "Answer: 4KDDZ6" is not in the correct format. The graph shows that the model already at rank 1 has learned to output the right format (a six character code). The remaining performance gap isn't helped by increasing rank. The red line means, given that the model outputted the correct format, is that output a valid answer within the dataset? In other words, does the output exist as a correct answer for some query in the dataset? The red line shows that the model constantly improves with rank on this. The green line means the share of answers that, given that the output is correctly formatted and exists in the pool of answers, were also correct for the specific query. Multiplying the values for Format (blue), Vocabulary (red), and Binding (green) gives the GIP accuracy of the system.

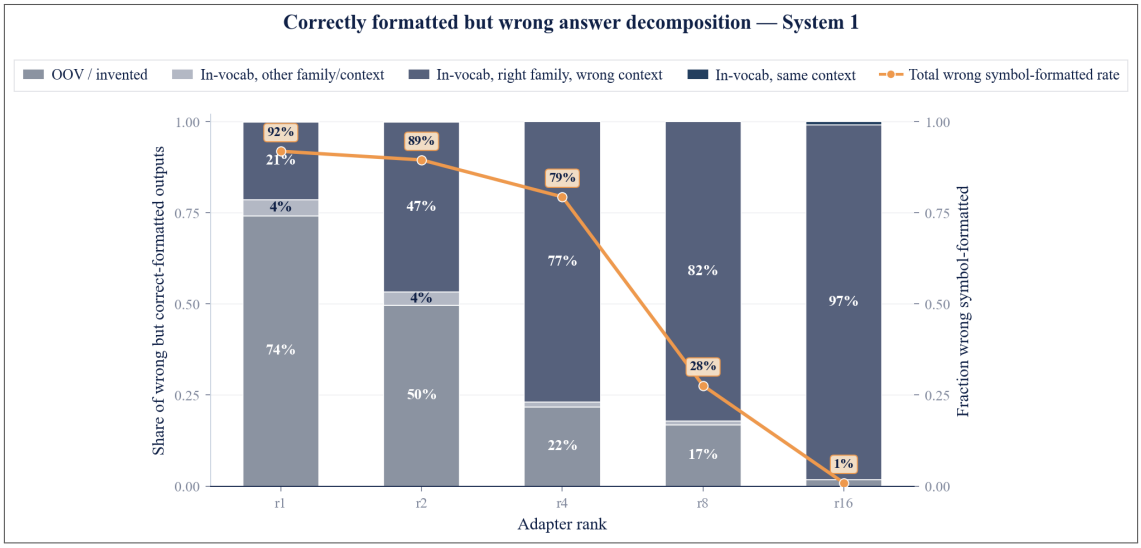

Figure 4.8: Correctly formatted but wrong answer decomposition for System 1 in the stress setting.

Figure 4.8 describes what types of mistakes the system makes. Dark grey means the share of wrong outputs that didn't exist in the answer pool - in other words, it was outside of the vocabulary. For low rank, this is the main issue, and it decreases with rank. Blue means the share of wrong outputs that exist in the answer pool, and belongs to the requested attribute, but was incorrect for that specific context: in other words, the output is indeed the value for that attribute, but for another context. This type of issue is the dominating problem for the larger ranks. Light grey is similar to blue, but it means that the value in fact belongs to a different attribute. Dark blue means the output belongs to the context in the query, but for another attribute (less than 1% of mistakes belong to this category for any of the ranks). Overall, the pattern is the following: for low rank, mistakes are usually due to the system making up answers, and for larger ranks, the issue is usually that the system outputs a real value belonging to the correct attribute but is associated to the wrong context. Moreover, the line chart indicates the total fraction of correctly formatted but incorrect answers, showing a clear decrease as rank increases.

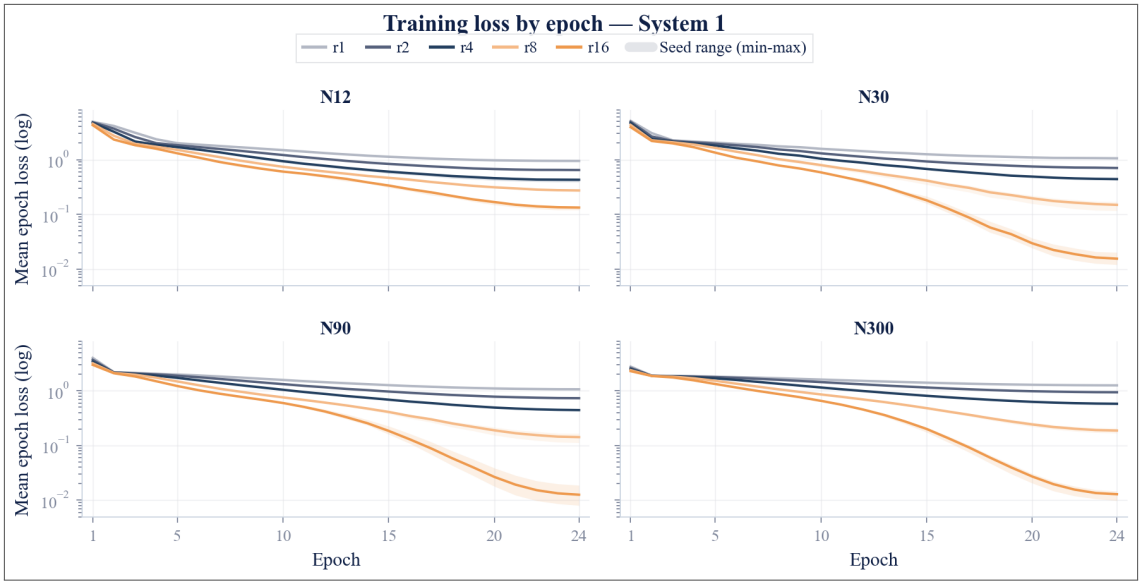

Figure 4.9: Training loss for System 1 on stress datasets across LoRA ranks.

Figure 4.9 shows the corresponding training behaviour to Figure 4.5, for the stress datasets. The same overall pattern holds: higher ranks achieve lower loss, and the gap becomes bigger as training progresses.

4.2 System 2

The following section reports the results from the System 2 experiments. We evaluate System 2 across different corpus sizes (N) and two levels of confusability: low confusability (S_{low}) and high confusability (S_{high}).

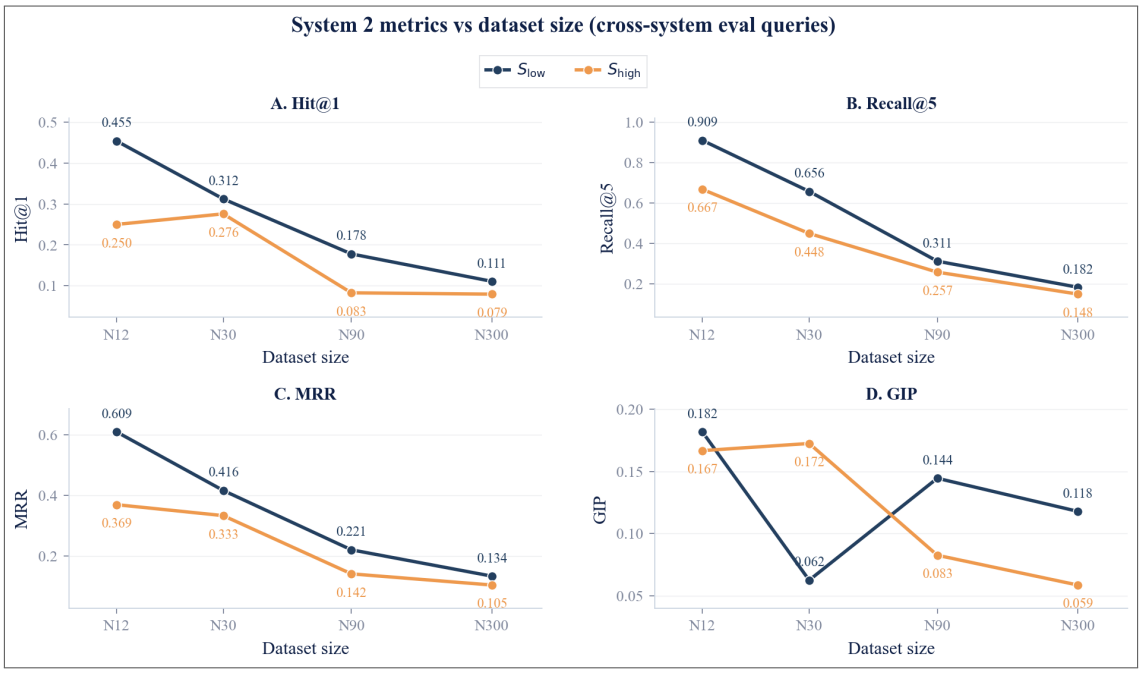

Figure 4.10: Retrieval and generation metrics across dataset sizes for System 2.

Figure 4.10 makes it clear that an increase in dataset size increases the difficulty both of retrieving the gold chunk and in generating a correct answer. Also, the S_{low} setting (lower confusability) generally performs better than the S_{high} setting (higher confusability). The largest difference between the two S-settings is seen in GIP for N300, where GIP drops more heavily from low to high S than the retrieval metrics do.

4.3 System 3

The following section reports the results from the System 3 experiments.

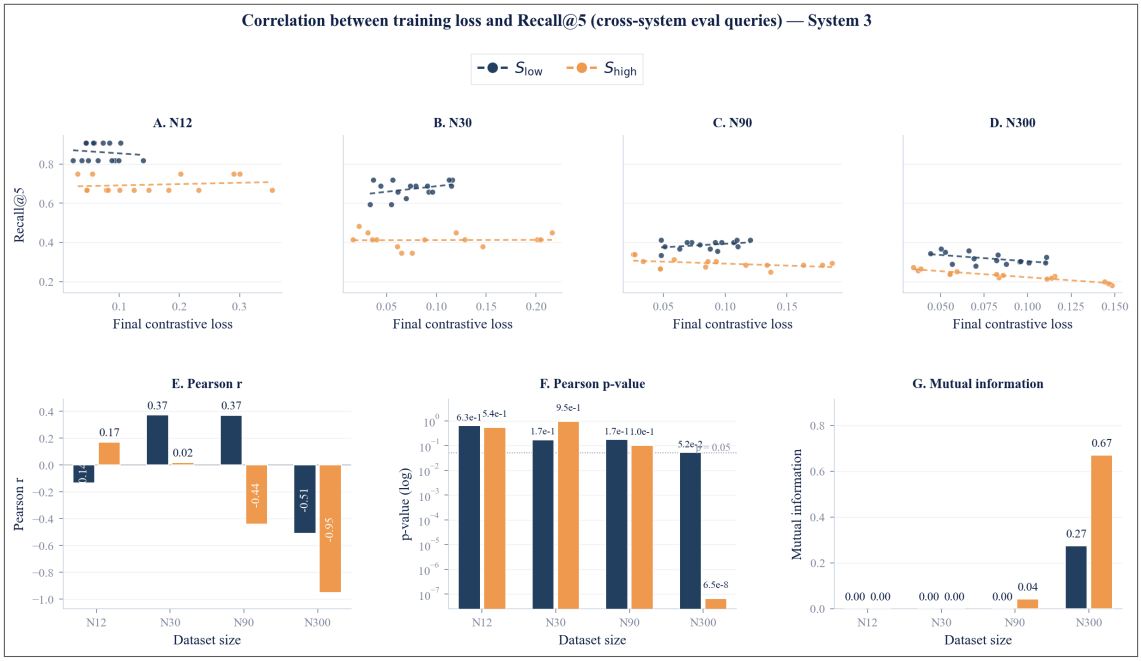

Figure 4.11: Correlation between training and retrieval during evaluation in System 3.

Figure 4.11 describes the correlation between the training loss and the Recall@5 metric. The scatter plots on the top show, for each dataset size, the correlation. Overall, these figures indicate that only the experiments on the N300 data seem to show an actual relationship between the training task and the evaluation task. This is because N300 are the first datasets with a convincing negative Pearson r, low associated p-value (way below 5%), and substantial Mutual Information. Therefore, the rest of the results described for System 3 focus on N300 data, since this appears to be the data regime where the LoRA training has clear correlation with retrieval performance.

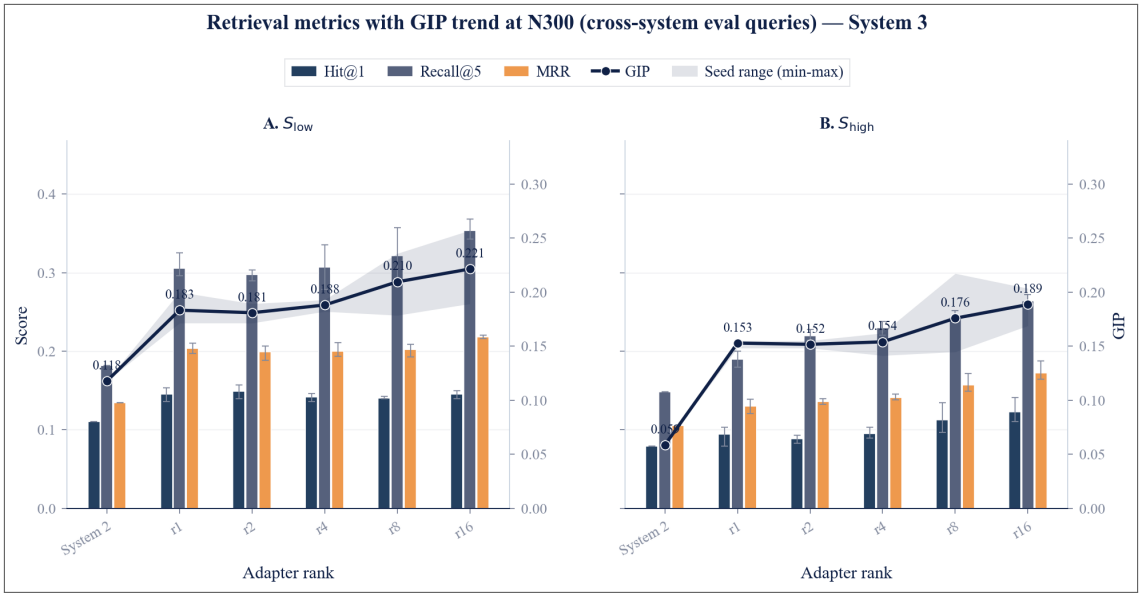

Figure 4.12: Performance metrics across ranks for System 3, N300 setting.

Figure 4.12 shows the Hit@1, Recall@5, MRR, and GIP for System 2 and ranks r1-r16 for System 3. Overall, Recall@5 appears to be the retrieval metric most improved by the LoRA training. Moreover, S_{low} leads to higher retrieval metrics than S_{high}.

GIP also increases with rank, and it appears to close the performance gap between S_{low} and S_{high} (since the System 3 GIP for S_{high} is much closer to the S_{low} performance than System 2). Moreover, GIP seems to follow Recall@5 relatively closely.

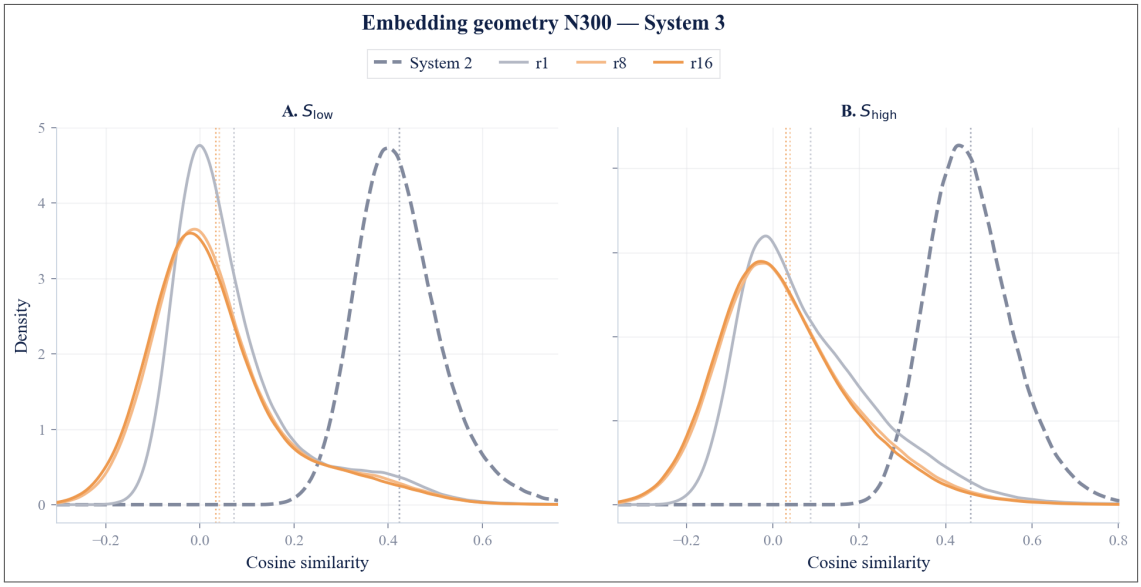

Figure 4.13: Embedding geometry shift for System 3, N300 setting.

Figure 4.13 shows that the embedding space is substantially changed by the LoRA training. The figure shows the density distribution of the cosine similarity for all

chunk pairs in the embedding space. The dotted line shows that for System 2, the embeddings are closer together and there's less spread. Training with rank 1 quite strongly moves chunks away from each other generally. This effect is shown to increase with increasing rank.

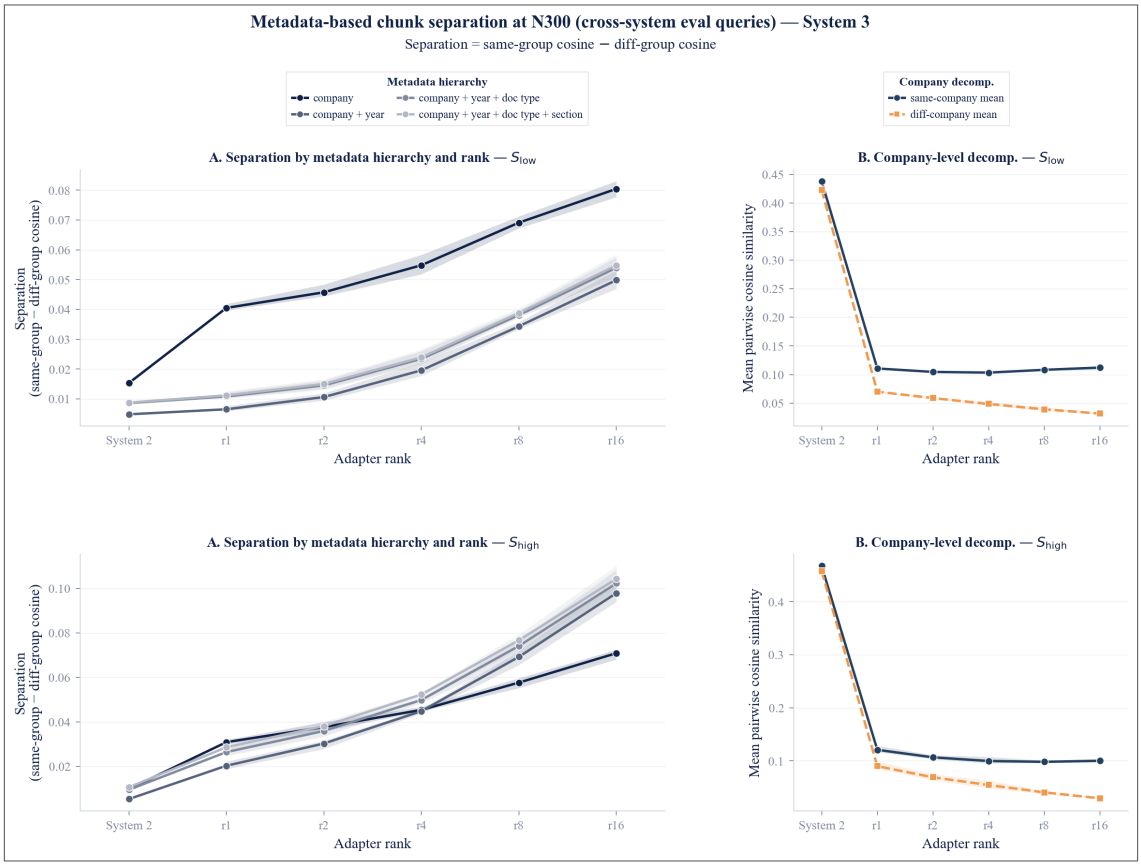

Figure 4.14: Context separation across ranks for System 3, N300 setting.

Figure 4.14 describes how the separation between chunk embeddings changes with LoRA rank, across different groupings. For example, the "company" line in Panel A means "the average cosine similarity between chunks belonging to the same company, minus the average cosine similarity between chunks belonging to different companies". The separation increases monotonically with rank. For S_{high}, the hierarchy between the most separated groupings also changes as rank increases. Moreover, in S_{low}, the company-separation is substantially stronger than the others, both at System 2 and across ranks in System 3.

Panel B looks deeper into the embedding space change for the company-grouping specifically. The blue line shows the average cosine similarity between chunks belonging to the same company, and the yellow one the average cosine similarity between chunks belonging to different companies. Both measures decrease drastically already at rank 1. However as rank increases, the diff-company mean decreases much quicker than the same-company mean (which stays relatively stable), explaining the increasing separation in Panel A.

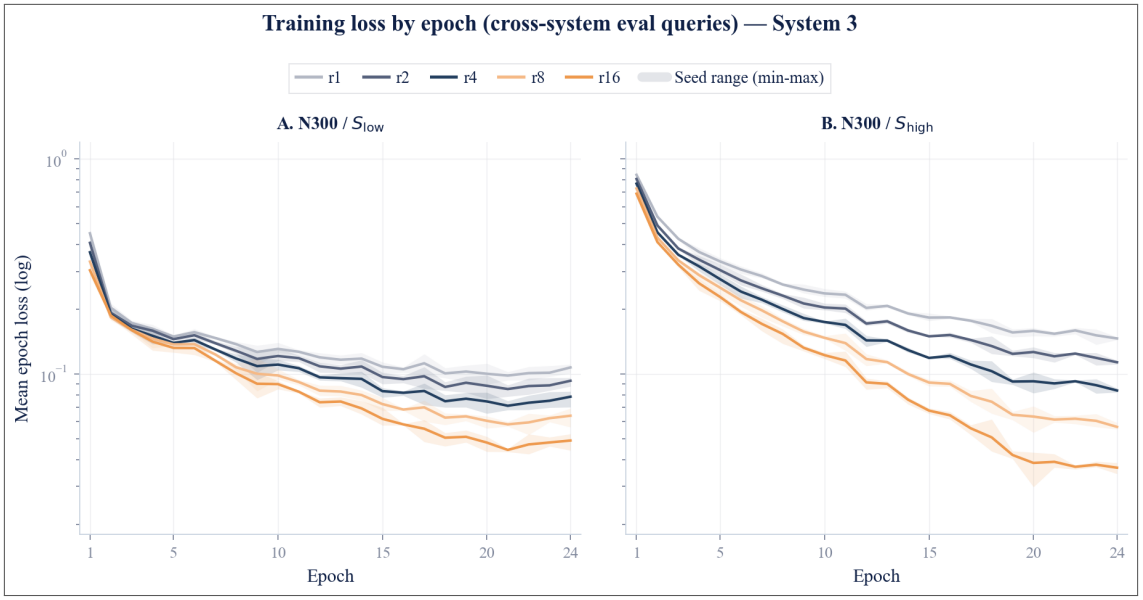

Figure 4.15: Training loss for System 3 across LoRA ranks.

Figure 4.15 shows the training loss for System 3 across LoRA ranks. The training curves show some noticeable fluctuations. Higher LoRA ranks achieve lower training loss, and the separation between final training loss across ranks is greater in S_{high} than S_{low}.

4.4 System 4

The following section reports the results from the System 4 experiments.

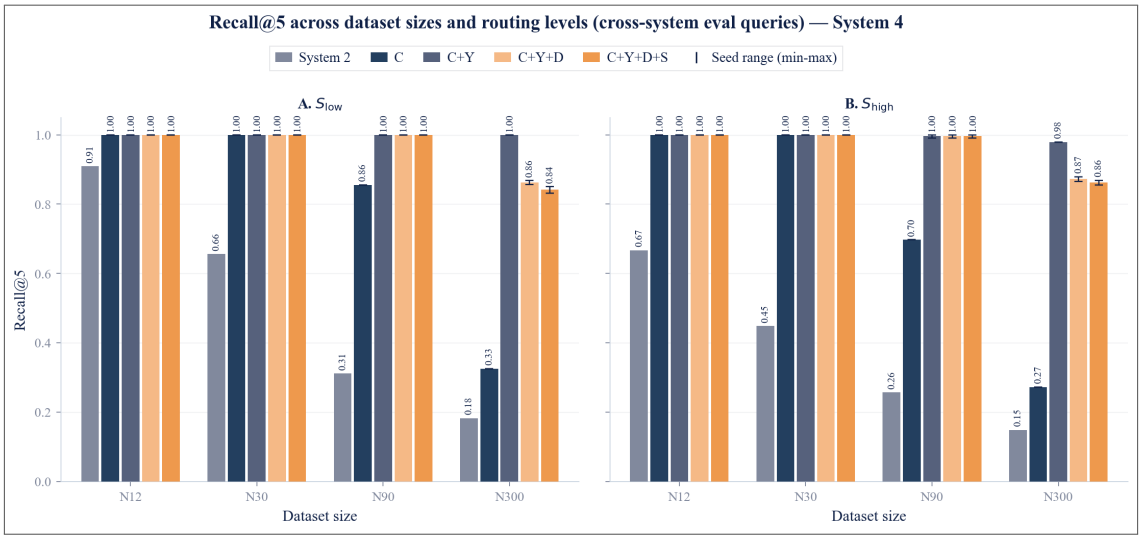

Figure 4.16: System 4 Recall@5 across dataset sizes and routing levels (r=16).

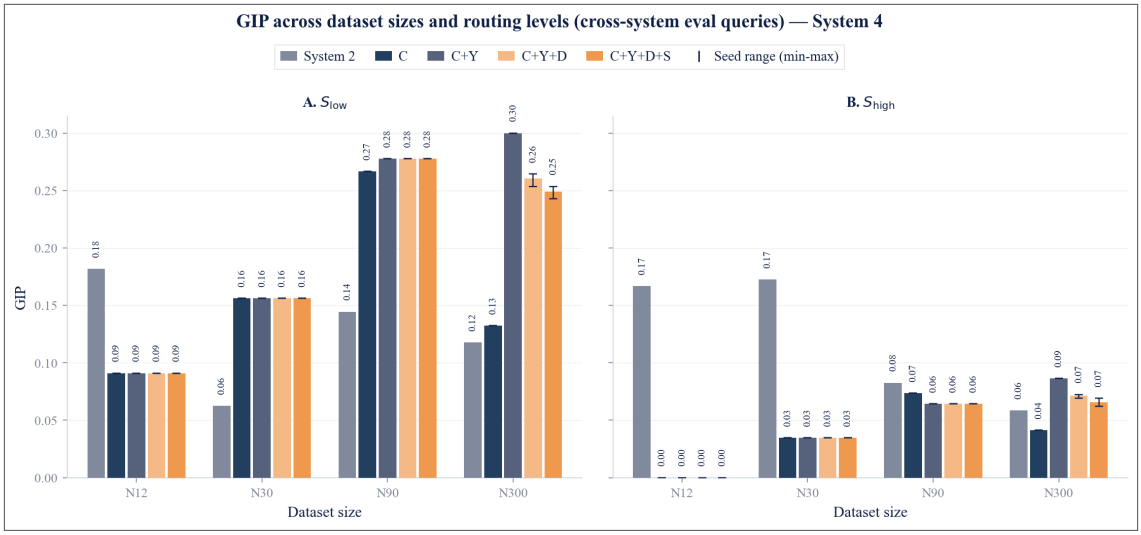

Figure 4.17: System 4 GIP across dataset sizes and routing levels (r=16).

Figure 4.16 and Figure 4.17 show how retrieval and generation performance varies over dataset sizes and routing levels. These results concern runs using LoRA rank 16. The results show that introducing filtering leads to a large improvement over the baseline System 2 retrieval across all configurations tested. The generator performance generally follows the retrieval performance, however the results for S_{high} show a much lower GIP score than S_{low} - an effect not seen in the corresponding retrieval metrics.

Since the results for the N300 datasets show the greatest amount of difference across routing levels, and since the System 3 results were primarily described for the N300 datasets, the remaining System 4 results focus on this dataset size.

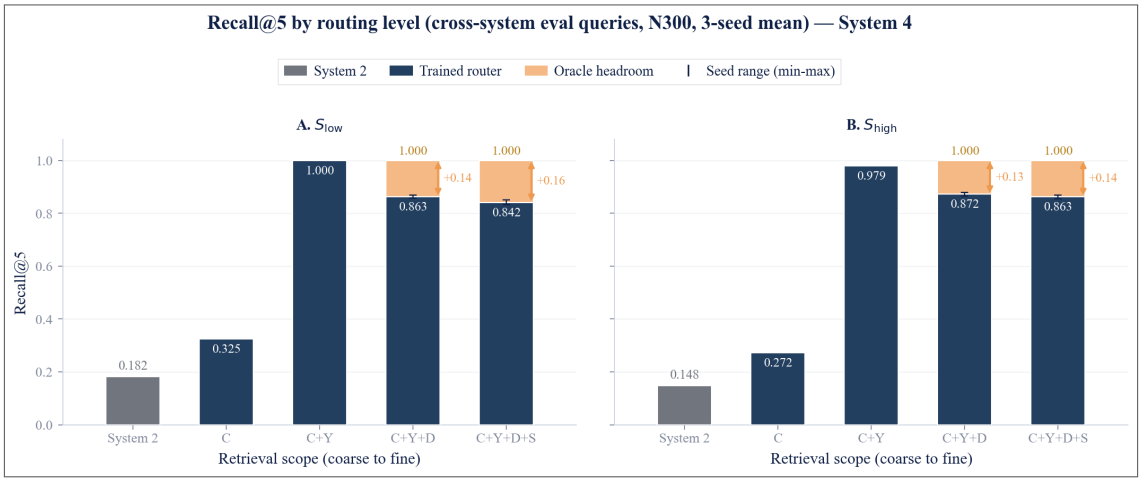

Figure 4.18: Recall@5 across different routing levels (r=16).

To understand the cost of routing errors, the achieved results are also compared against the performance achieved when using an oracle router that always routes retrieval to the correct context. This is shown in Figure 4.18. The oracle system achieves higher performance at finer routing levels.

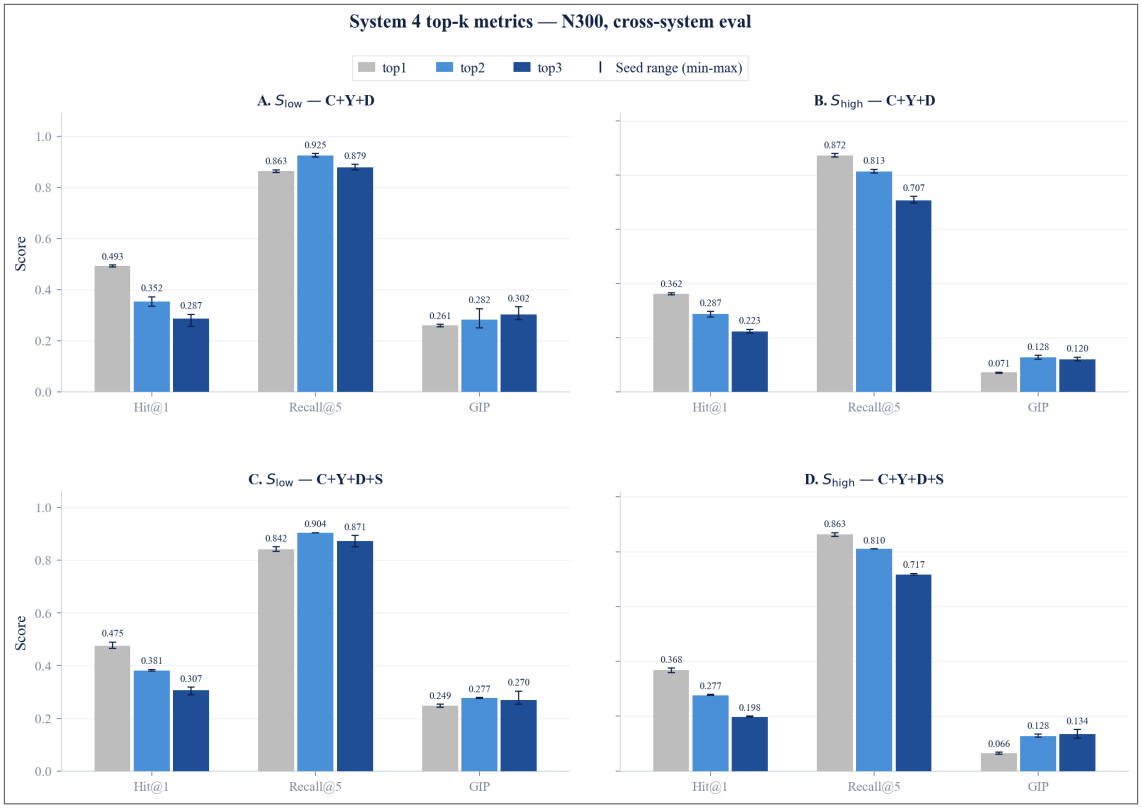

Figure 4.19: Comparison in Hit@1 between using multiple routing candidates (r=16).

Figure 4.19 shows the effect on retrieval and generation performance of allowing retrieval on the union of the chunks belonging to the top-2 and top-3 predicted contexts, for the two more granular routing levels C+Y+D and C+Y+D+S. Hit@1 is hurt by it, and Recall@5 is slightly helped by it in the S_{low} setting but hurt by it in the S_{high} setting. For the generator performance, there's a certain benefit to increasing the filter to include the top-2 or top-3 predicted contexts, and for the S_{high} setting (which was noted in Figure 4.17 to be a much more difficult setting for the answer generator) the benefit is more visible.

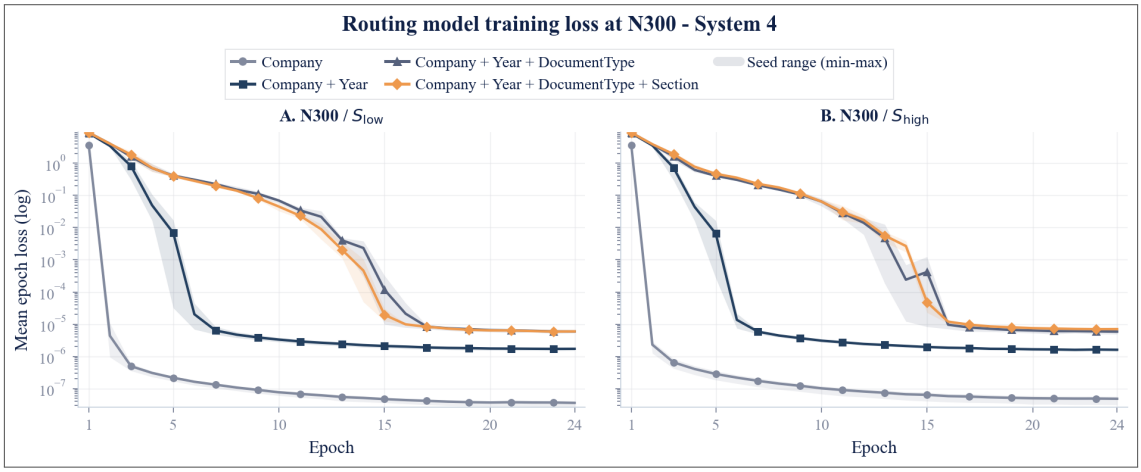

Figure 4.20: Training loss for System 4 across routing levels (r=16).

Figure 4.20 shows the training loss for the router. The loss decreases across epochs which indicates that the router is learning the classification task. Simpler routing tasks (e.g. Company) converge quickly while more complex routing levels show much slower and less stable convergence.

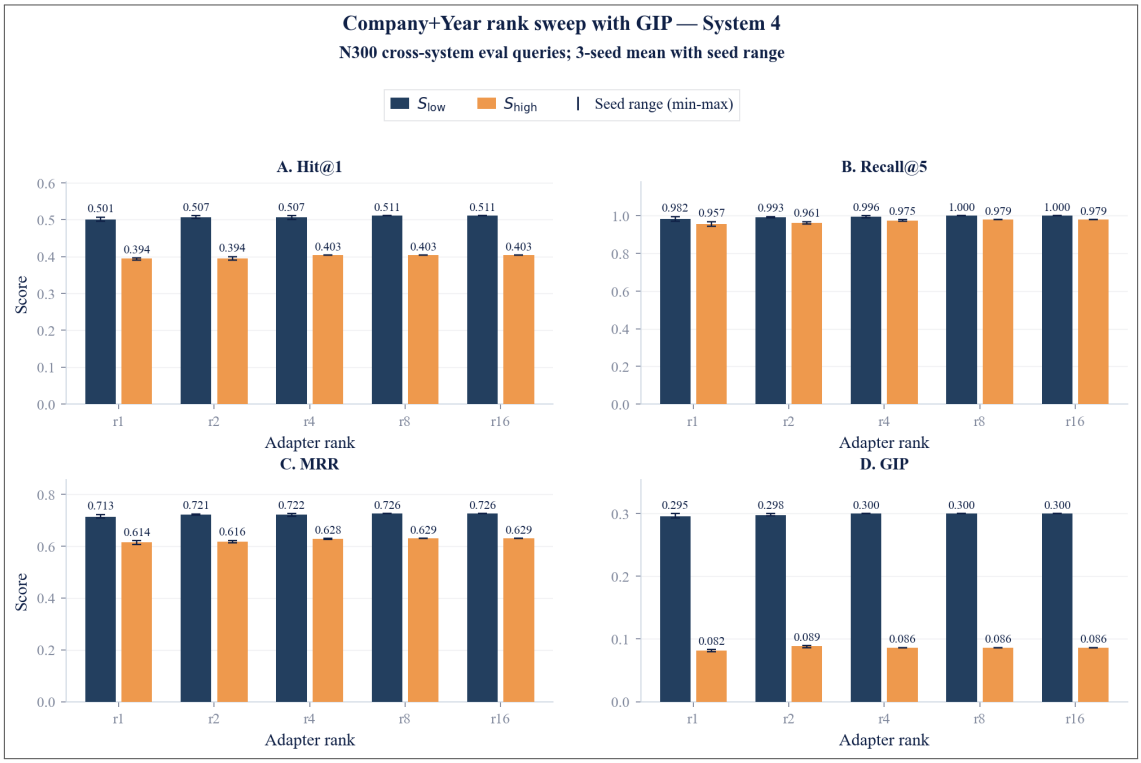

Figure 4.21: Retrieval and generation performance for the rank sweep for the Company+Year filtering level on N300.

After observing strong performance for the Company+Year filtering level in the cross-routing level performance comparisons, additional experiments were conducted to evaluate how sensitive this setting is to the choice of rank. Figure 4.21 shows the results across ranks on the N300 datasets. The results indicate that the performance is relatively unchanged across ranks for this configuration.

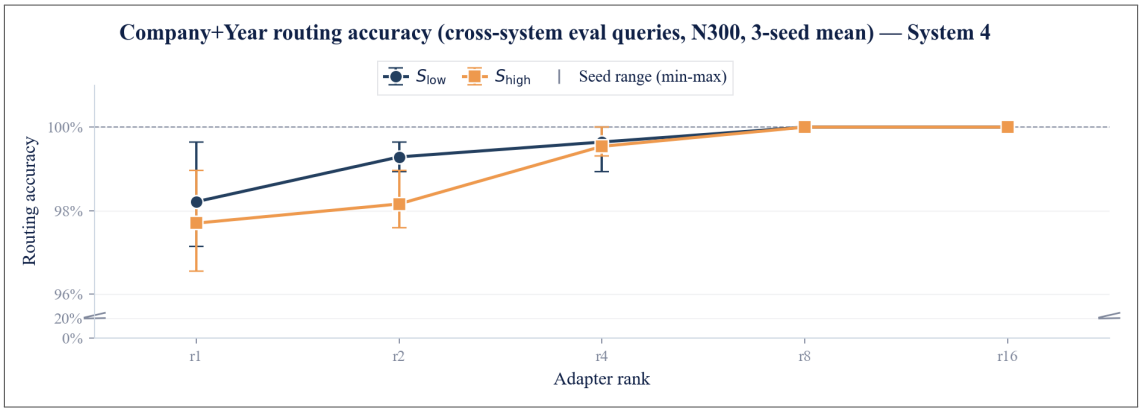

Figure 4.22: Routing accuracy for the rank sweep for the Company+Year filtering level on N300.

Tying into the results shown in Figure 4.21, Figure 4.22 shows that for the experimental setup, the system reaches almost 100% routing accuracy already at rank 1 for the Company+Year routing level.

4.5 System comparisons

The following section reports cross-system comparative results. Since the systems differ in where LoRA is applied and what training problem it is used to solve, the comparisons should not be interpreted as comparing absolute system quality. In particular, System 1 was evaluated on the same facts it had been exposed to during training but in different wordings, whereas System 3 and System 4 were evaluated on new fact queries they had not been exposed to during training - a difference emanating from the inherent system designs. System 1 also received, effectively, twice the number of training steps. The reader is thus encouraged to focus on the most important patterns which are how performance changes with rank, dataset size, and confusability setting - rather than focusing on absolute performance metrics in isolation.

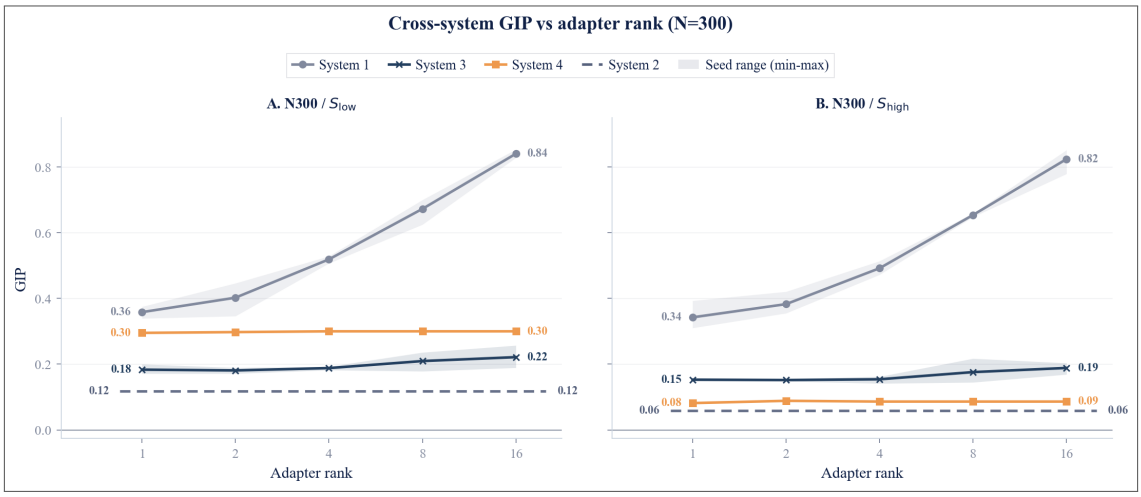

Figure 4.23: GIP performance increase comparison across systems as rank increases, N300.

Figure 4.23 shows, for S_{low} and S_{high} respectively, how the GIP performance gain changes with rank across systems. The comparison is mainly intended to show how GIP is affected by increasing rank in the three LoRA roles tested, rather than for strictly comparing the absolute performance of each system. System 1 shows the largest GIP gains as rank increases. System 2 stays the same, as it isn't trained with LoRA. System 3 increases gradually and modestly. System 4 with the Company+Year routing level increases barely anything with increased rank.

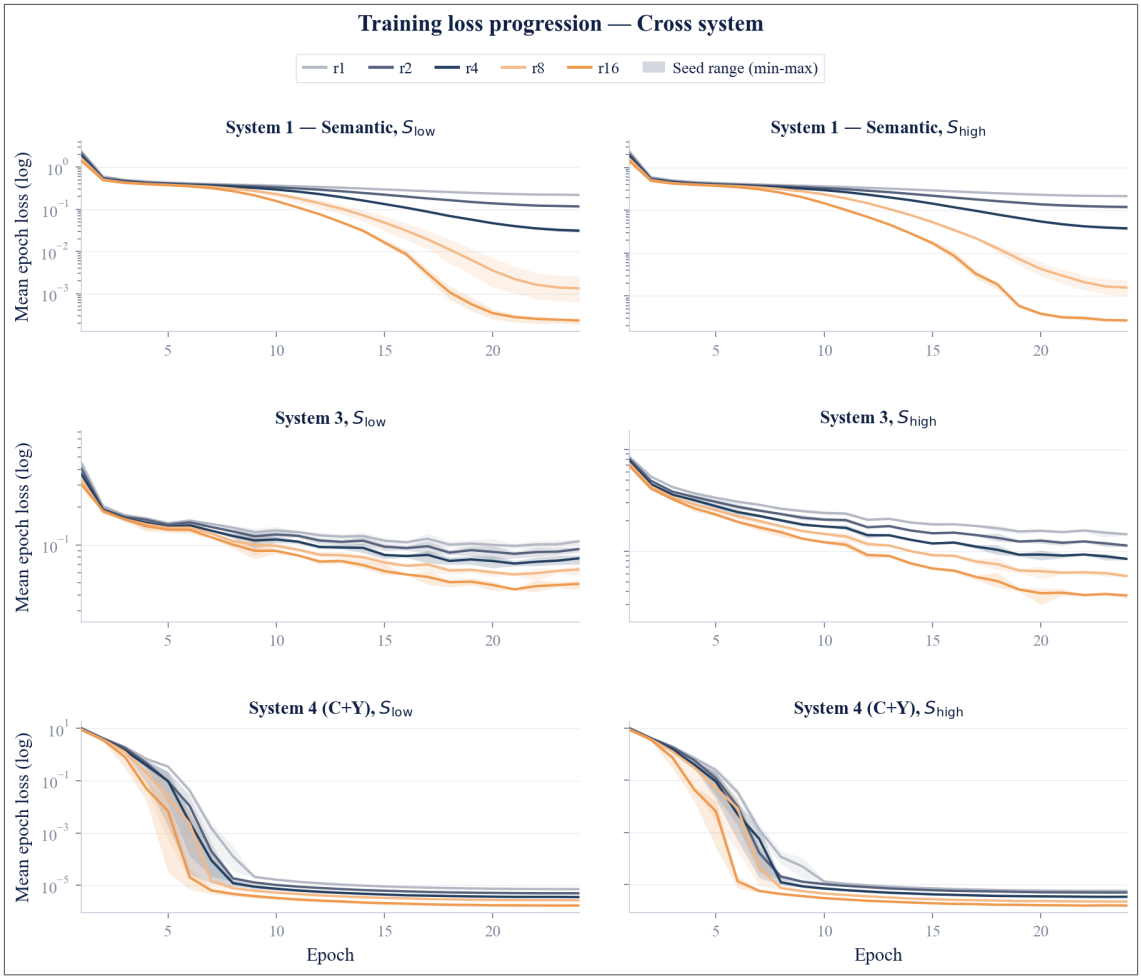

Figure 4.24: Training loss curves for each of the trained systems, N300.

Figure 4.24 compares the training loss convergence across systems for the N300 dataset size, with S_{low} on the left and S_{high} on the right. The System 4 curves are for the Company+Year routing. The x-axis shows the training progress, the y-axis shows the loss. The general pattern is that System 4 converges the quickest, with System 3 converging the slowest and System 1 in between.

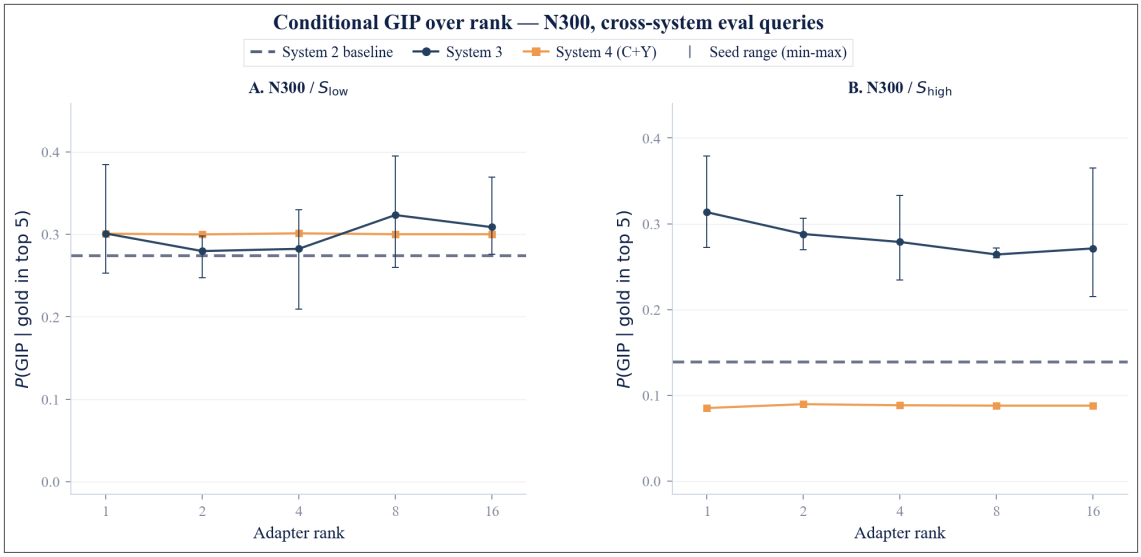

Figure 4.25: P(correct answer|gold in retrieved chunks) as LoRA rank grows for Systems 3 and 4 across all dataset conditions. The dashed line indicates the System 2 baseline.

To better understand how the RAG-based systems utilize the retrieved chunks, Figure 4.25 plots the share of test queries where the generator answered correctly when given the gold chunk among the retrieved chunks. For S_{low}, the three retrieval systems have roughly similar conversion rate, whereas for the S_{high}, System 3 remains at the same rate whereas System 2 drops, and System 4 drops even further.

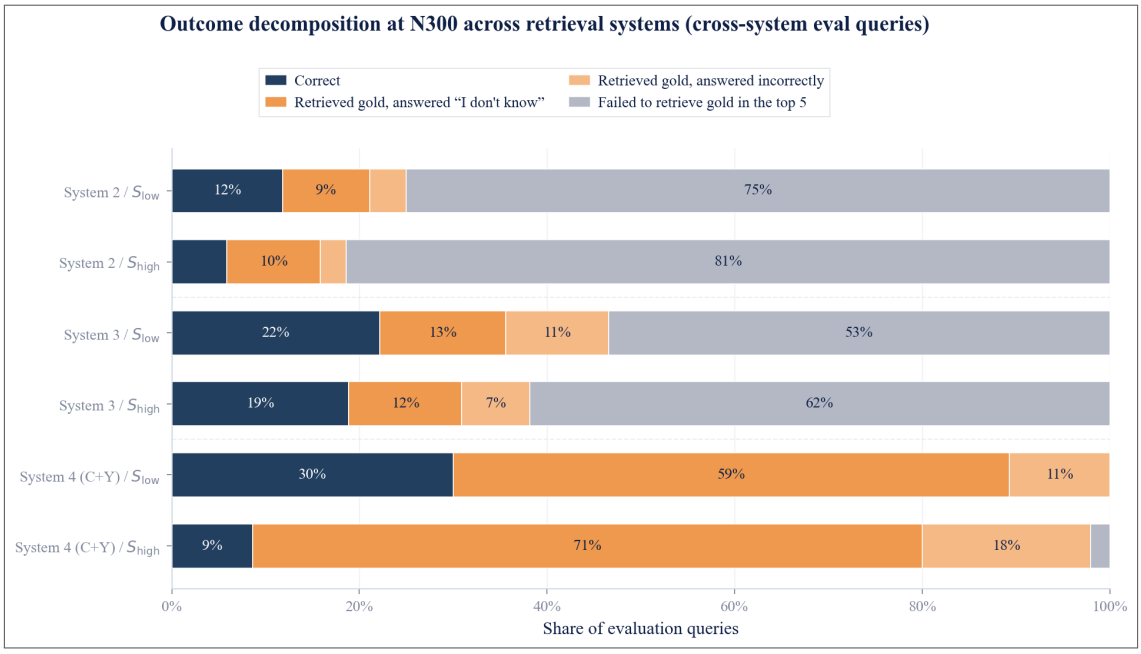

Figure 4.26: Outcome decomposition across retrieval systems at N300.

Figure 4.26 describes the differences in what kinds of output each retrieval system gives at S_{low} and S_{high}. For System 2 and System 3, the main driver for incorrect generation is retrieval failure. For System 4, it is instead a failure to convert the retrieved gold chunk into a correct answer.

5
Discussion

The results show that LoRA can be used to improve question-answering systems in all three tested roles. The following chapter discusses the observed patterns, interprets them, and relates back to the research questions.

5.1 System 1

System 1, being the purely parametric approach explored in this report, is intended to help answer the question regarding how well LoRA can internalize structured knowledge in its adapter weights. The two different (synthetic) data regimes tested - the semantic data (being the shared eight datasets), and the stress data (the symbolic data stripped of all semantics) - both show different aspects of the nature of LoRA in this application.

In the semantic setting, performance is shown to improve with larger data sizes in the regime investigated. Fact exposure is kept constant across experiments since the system sees any given fact twice during training, which has the effect that the model gets more optimization steps with larger datasets. Increasing the dataset size therefore has two simultaneous effects in this setup: it increases the factual load, but it also increases the exposure to similar semantic topics and similar structures (for example, the model sees similar query formulations and connections between words like "revenue" and "sales" more often). In the semantic data setting, the latter appears to have a much stronger effect on performance than the former, since performance increases with dataset size.

Moreover, the results on the semantic data show that rank has a tangible effect on the system's ability to truly bind a fact within the adapter parameters. The ability to remember the answer to a seen query reaches high performance for low ranks, but the ability to generalize across different formulations of the same query requires much higher rank in this setting. This is an empirical indication that learning an abstract internal representation of a fact requires a higher adapter space dimensionality than simply binding specific formulations to answers. Moreover, unseen-fact performance also increases slightly with rank which is interesting since the system shouldn't reasonably have a way to generalize to unseen facts. The reason for the increase may be that the model gets better at giving the right answer format and thus more often guesses correctly, or perhaps there are subtle patterns in the data that the model can learn at higher ranks that also improve guesses.

For the stress data, certain other interesting patterns arise. There are two phenomena that differ starkly from the semantic results. Firstly, there seems to be a performance threshold between rank 4 and rank 8 in the tested stress data regime, where performance takes a leap - very different from the semantic results, where performance gains from rank were more gradual. This suggests that when semantic meaning is removed, LoRA must store arbitrary bindings which has a clearer rank requirement than memorizing the semantic facts. Secondly, the results show that for ranks 1-4, increasing data size hurts performance. Moving from the datasets N12 to N30 seems to help, but then, performance decreases. This effect is not present for ranks 8 and 16. The implication is that for lower ranks, this memorization-heavy stress setting leads to true parameter saturation - the LoRA adapters simply can't fit the necessary bindings when the number of facts increases. This interpretation is reinforced by the fact that in the stress regime for the lower ranks, larger datasets lead to larger training loss, an effect which is flipped when moving from rank 4 to rank 8 (see appendix Figure B.4).

Another interesting observation is that the system learns different "levels" of the stress learning task more or less well, each of which requires different rank sizes to achieve. The results show that low rank adapters are capable of learning the format of the task, but not the factual content. Then, as rank grows, the adapters start to learn the answer space - the existing answers are encoded into the parameters, but the binding between query and answer is still weak. Then, the big leap comes at rank 8, where the adapters also connect the right answer to the right query. Connecting to this, the results also show that the major change in what types of mistakes are made, is that for lower ranks, mistakes are usually due to inventing answers, and as rank grows, mistakes are more often existing answers but tied to the wrong query - aligning well with the findings that internalizing knowledge happens in stages. In all, these figures show that LoRA rank controls not just how many facts the system can recall, but at which level the system has learned the data. Therefore, the results of System 1 indicate that using LoRA as parametric memory may be a particularly promising approach when the task is just to learn expected formats, or to learn a set of things without having to tie each entity to another entity through a relationship.

The results also show that the LoRA adapters use more effective dimensions as the rank increases, but there are some differences across the semantic and the stress regimes. There is clearly a correlation between effective rank and performance, but there's still spread in the performance that isn't explained by the effective rank. This makes sense in the broader context of our analysis of the semantic data results: the model benefits very clearly from the increase in data size which we argued is due to the increased exposure to the task. In the stress regime, the same spread is seen for the larger ranks r8 and r16, but for the smaller ranks where the saturation effect was observed, effective rank seems to explain a larger portion of the performance. Moreover the following pattern is observed for both regimes: as rank increases, there's a small number of very important update directions, and the additional directions provided by higher rank act more like small refinements.

One hypothetical interpretation of the stress results with the steep threshold between rank 4 and rank 8 is the following: the first update directions are used to encode the expected answer format, the next few added directions are used to encode the answer space, and finally when rank reaches 8, the small updates in the additional directions are made to tie each answer in the answer space to a concrete context which enables correct question-answering.

5.2 System 2

The results of the RAG baseline System 2 show that both retrieval performance and generator performance decrease as size increases and as confusability increases, aligning with expectations. As dataset size increases, the retriever must rank the gold chunk against more and more distracting chunks. As S increases, the distracting chunks become increasingly similar to the gold chunk and the gold chunks become less explicit. The gap between S_{low} and S_{high} is largest for small datasets and becomes smaller as N increases. One possible explanation is that when the dataset is small, differences in confusability have a stronger effect because of the fewer overall chunks. As the dataset grows the overall difficulty of retrieval increases. This in turn leads to both settings(S_{low} and S_{high}) starting to perform poorly, which reduces the relative difference between them.

5.3 System 3

System 3 builds directly upon System 2, using LoRA to adapt the embedding space. It is important to remember that the training task is to become better at discriminating between the gold chunk for a given query and randomly chosen wrong chunks. As opposed to System 1, it was evaluated on facts that had not appeared in training queries.

A notable finding is that clear correlation between training loss and Recall@5 was only found for the largest dataset size setting, N300. This could be because in the smaller datasets, there may be too few confusable chunks for the contrastive learning to result in a meaningfully improved embedding space geometry. At N300, retrieval may become difficult enough that training actually helps. It could also be that larger datasets are required for the true improvements to show - the smaller datasets may be noisy since there's fewer test chunks to evaluate on. Nevertheless, the correlation analysis shows that training does have a measurable effect on the system's ability to rank the gold chunk in the top 5 returned chunks, at least for the larger datasets, which is where the discussion focuses.

The results show that the LoRA training has a larger beneficial effect on Recall@5 than Hit@1 which means that the model becomes better at placing the gold chunk in the top 5 than it becomes at placing the gold chunk at specifically ranking 1. This is consistent with the way System 3 is trained with contrastive learning; each

query-chunk pair comes with four uniformly sampled negative chunks. These negative chunks are likely to be quite unrelated to the query which makes the training problem, at least in part, the task of separating clearly relevant chunks from clearly irrelevant chunks (separating in terms of cosine similarity). In evaluation, retrieval is exposed to less obvious negatives - chunks that perhaps share the same company, topic etc. as the gold chunk. The model is thus trained on an easier discrimination task than what leads to Hit@1 performance.

The results also show that LoRA greatly changes the geometry of the embedding space, making chunks become more spread out, in a way that supports this interpretation. This is achieved during training by pulling chunks away from each other, particularly pulling chunks from different "groups" (in terms of metadata) away from each other. Tying into the fact that Recall@5 is most helped, it may be that this type of "spreading chunks apart" from each other is a good way to separate the gold chunks from the clearly wrong chunks, but it may not be fine-grained enough to enable maximum benefit to Hit@1.

The geometry analysis of the embedding space shows that in the S_{low} setting, the company-separation is the strongest separating grouping, which is the most striking thing one sees when looking at the left panel in Figure 4.14. A possible reason for this is that in S_{low}, "company" is already the most powerful separator, since chunks in that setting belonging to the same document concern visibly different topics, whereas chunks from different documents but for the same company use a clear "style". At S_{high}, chunks within a document share very similar wordings, more so than they share with chunks from different documents - this makes the "same exact document" the strongest separator, which is what Figure 4.14 indicates. LoRA training in System 3 appears to amplify whatever grouping is already the strongest.

Finally, concerning the adapter analysis for System 3, the results show that the effective rank and spectral compression follow a similar pattern as observed for System 1. However, the interpretation is different. For System 3, the effective rank affects how flexible the embedding space reshaping becomes, rather than how many fact bindings can be stored. This therefore gives a direct indication that although the LoRA task is fundamentally different between these two systems, the adapters still show similar structure in terms of dimension usage and singular value magnitude distribution.

5.4 System 4

System 4, similarly to System 3, builds directly on System 2 but does so by adding a filtering step before retrieval. Like System 3, it was evaluated on facts that had not appeared in training queries.

At a high level, filtering is clearly helpful for retrieval, which suggests that a big part of the difficulty in System 2 comes from having too many irrelevant candidates and that the correct chunk becomes much easier to find when the search space is

reduced. The results show that very low rank suffices to achieve almost perfect routing accuracy at the Company+Year level, a finding that should be understood within its context. The company and year are usually named explicitly in the test queries, so classification at this routing level is to be expected to be quite easy. The low rank requirements should therefore not be interpreted as a general finding for all routing settings - evidently, routing accuracy decreases once the more difficult fine-grained routing is used and the classification task thus becomes more difficult.

Context routing introduces a new type of error that does not exist in System 2 or System 3. In those systems, the correct chunk is always somewhere in the search space. In System 4, it can be completely removed before retrieval even begins which is observed at finer routing levels when routing fails. This implies that routing level should be considered an important design choice depending on the context that the system is applied to, since the best performance is achieved when filtering is strong enough to remove irrelevant information but not so strong that it risks removing the correct answer completely.

A key finding is that strong retrieval performance doesn't necessarily translate to strong generation performance in System 4. This is made clear by how much GIP drops when moving from S_{low} to S_{high}, while retrieval performance doesn't decrease nearly as much. A large portion of all retrieval-successful queries still result in an "I don't know"-answer in the S_{high} setting, which is less common in S_{low}. The reason for this may lie in the differences in the S-regimes. In S_{low}, chunks are more explicit, for example if it concerns the segment Middle East, then "Middle East" often occurs in the text whereas in S_{high}, it's more often an opaque description like "a core customer segment". System 4's routing constrains retrieval to chunks from the gold context, so the retrieved five chunks are usually the gold chunk and other paragraphs from the same context (depending on router granularity, for example the same Company+Year). In this document collection, when the gold chunk doesn't mention the identifier explicitly, usually the rest of that document doesn't either. Therefore the generator chooses to answer "I don't know" since that option is encouraged in the prompt if the generator is uncertain.

Another important result is the effect of using multiple routing candidates. Allowing the system to consider the top-2 or top-3 routes reduces the impact of these routing errors that have been encountered and improves Recall@5 for the N300 S_{low} dataset. This can be understood as a compromise between strict filtering and global search. Instead of committing to a single path the system keeps a few options open. But this also reduces Hit@1 since the search space becomes larger again. However, while retrieval is generally hurt by using top-2 or top-3 routing, generation is actually helped. This connects back to the previous paragraph discussing how good retrieval doesn't always lead to good generation. The implication is that generation is sensitive to the set of retrieved chunks, not only to whether the gold chunk is present. By including additional routes, the system may include a more diverse set of chunks in retrieval, that don't all concern the gold chunk's company and year, but instead may help the generator infer the correct answer type.

Regarding the effective rank of the LoRA adapters in System 4, similar patterns are observed as for System 1 and System 3, further indicating that the adapters evolve similarly as rank increases for all three applications. The big difference is that there's no real correlation between effective rank and GIP in System 4 at the C+Y level, which can be explained by the fact that the training task is essentially solved already at rank 1.

5.5 System comparisons

The four systems differ in their training objectives, the retrieval stage, and how they are expected to generalize. Moreover, the systems weren't optimized in terms of setting details and improvements in absolute performance may well be possible by adjusting certain settings. Comparisons across the systems should thus not be seen as a ranking of overall system quality. Instead the interesting question is how each LoRA role behaves with respect to rank, dataset size, confusability, retrieval performance, generation performance, and practical implications.

Rank sensitivity and direct question-answering performance. The results show that System 1 achieves the largest GIP performance gains when rank is increased. This could be expected since GIP is a generation metric - when considering the question-answering pipeline, System 1's placement of LoRA is the closest to generation and is trained directly on query answer generation, on the same facts as in evaluation, but with different wordings. System 3's and System 4's improvements are modest in comparison, but they stem from fine-tuning at the earlier stages in the pipeline which means LoRA improves answer generation only indirectly in those systems. While System 1 shows large GIP improvements with increased rank, System 3 shows smaller ones which are driven by improved retrieval performance (indicating that embedding adaptation is a slower way to improve generation performance), and System 4 shows no real gains at all at the semi-coarse routing level Company+Year since the routing at that level was basically solved. However this also means the main benefits from System 3 and System 4 come at an earlier rank than for System 1, although the training task for System 3's embedding adaptation appears slower to converge and less smooth in doing so.

Here it is also important to stress the comparison confound regarding optimization steps. System 1 being trained for twice the number of optimization steps certainly helps it, and even though it is exposed to each training example the same number of times as the other systems, this should be considered an "unfair" advantage for System 1. When looking at the loss progression, System 4 at C+Y routing level clearly converges with the available optimization steps (this is also true for the finer routing levels although to a less obvious extent). For System 3, some convergence can be sensed but it is likely that retrieval performance would have improved further had the system received the same number of optimization steps as System 1. However this would introduce another confound in that System 3 would be exposed to the same number of training examples twice as often as System 1. The most

fair approach would have been to only train System 1 on one query per fact rather than two, which most likely would decrease the total GIP performance of System 1 although similar rank and data size scaling patterns would be expected. Again, this places further emphasis on the need to avoid comparing absolute performances across the systems, and instead focus on the observed patterns and how they differ.

Retrieval quality versus generation quality. The results also highlight that improving retrieval does not always entail equally improved generation. System 4 achieves substantial retrieval performance gains by filtering out chunks from incorrect contexts before the retrieval stage, but these gains are not always translated into correct answers. This is especially clear in the S_{high} setting as shown in Figure 4.25, where both System 2 and System 4 drop in their ability to convert retrieved gold chunks into correct answers, but System 4 drops the most. The reason is believed to be that the routing, in the tested experimental setup, constrains the retrieved chunks to a narrow context where, if paragraphs are too opaque in their wordings, the generator may choose to abstain from answering even if the answer is among the chunks retrieved. System 3 behaves differently, since it performs a global retrieval and allows retrieval of a more diverse set of chunks than System 4. System 3 actually shows an interesting strength in that it maintains its conversion rate from retrieval to generation performance (see Figure 4.25). The reason is believed to be that in some S_{high} situations, non-gold retrieved chunks can still mention identifying terms (for example the customer segment in question), which helps the generator to become confident enough to answer. This effect is exemplified in Figure B.12 in the appendix. This can explain why System 3 has a better conversion rate from retrieved gold chunk to correct answer than System 4, and it indicates that if S_{high} contained more explicit mentions of identifying terms, then the observed effect with poor conversion from gold chunk retrieval to correct answer generation would likely be smaller.

While this phenomenon is partly specific to this particular experimental setting, it still highlights a potentially elusive issue practitioners may encounter when filtering the retrievable chunks, depending on their data and overall system design. It's also empirical evidence that generation quality is dependent not only on whether the gold chunk is present among the retrieved chunks but also on how well the total set of retrieved chunks informs the generator.

Flexibility, scalability, and practical trade-offs. When considering deploying the three roles of LoRA tested in this thesis in real-world applications, one needs to also consider practical implications and characteristics of each role. One such aspect concerns how each of the systems tested would behave as new data is added to the knowledge base. System 1's strength in how it fine-tuned the generating model on the fact-level then becomes its weakness, since System 1 would need to be entirely retrained when new facts are added. System 3 isn't as brittle, since the system would still have a very fair chance to retrieve the added chunks - an effect of the RAG architecture. However, the system's benefits may diminish, especially if the added data differs greatly from the original data. For System 4, no retraining

should be necessary as long as new data belong to the original contexts - however, when new contexts are added, the router would need retraining (although since the routing task proved to be quite an easy classification task in training, this retraining would likely be more lightweight than for the other systems).

This creates a practical trade-off concerning the flexibility and scalability of the systems. System 3 was shown to be robust in how retrieved chunks are turned into correct answers, and the system would continue to function as more data were added - however, the training effort required is directly tied to the number of chunks in the vector database. System 4 on the other hand may be more efficient in the case where the number of new chunks grows much faster than the number of contexts, since the routing task depends on the context labels rather than the chunks directly (this is inferred from the system design and wasn't tested in the experiments). Then again, this efficiency comes with some brittleness: routing errors can lock out retrieval from finding the gold chunk, the system requires available and well-behaved metadata, and narrow routing has shown to sometimes lead to unexpected issues in generation even when retrieval is improved.

The comparison across the systems suggests that they should be regarded as complementary applications of LoRA rather than one being clearly superior. The systems are not mutually exclusive - all three roles may be incorporated into the same system, and the findings suggest that a promising direction for future work and practical implementation may be a hybrid system. In such a system, routing can reduce the retrieval space relatively coarsely, embedding adaptation can improve the vector space that retrieval is routed to, and parametric memory can be used for e.g., facts that rarely change or for format expectations.

5.6 Answering the research questions

Based on the discussion of the project's results, the research questions may be answered as follows.

1. What are the benefits and limitations of LoRA when applied in a question-answering system as parametric memory, when used for retrieval adaptation, and when used for context routing?

LoRA is useful in all three tested roles, with different strengths and weaknesses. The results do not identify a universally superior LoRA placement in a question-answering system. Instead they show that the usefulness of LoRA depends on the situation. In a semantically meaningful setting with a knowledge base size in the realm of a few thousand facts, LoRA as parametric memory appears to give the largest gains in question-answering performance as rank grows in the particular seen fact/new-formulation setting. However if the data lacks semantic meaning, LoRA as parametric memory appears less powerful. Moreover, despite the promising performance of LoRA as parametric memory, the lack of flexibility due to the need of retraining as new facts are introduced poses a downside to this approach.

As a tool for retrieval adaptation by fine-tuning the embedding model in a RAG system, LoRA shows less dramatic benefit than as parametric memory in the studied setting, and appears most useful for data sizes in the realm of a thousand chunks or more. However, it shows robust performance improvement across both tested levels of confusability, and has scalable potential as retraining isn't immediately required as new facts are added. Overall, using LoRA for retrieval adaptation shows promising performance for RAG-based systems.

As a tool for context routing in a RAG system, LoRA is shown to enable strong, low-rank performance gains when implemented as a multi-class classifier, requiring much less training effort and capacity than the two other approaches when using a relatively coarse filtering level, although with a lower performance ceiling than when using LoRA as parametric memory. Retraining is only required as new facts with new contexts are added. A potential drawback to keep in mind is the potentially unpredictable effects on generator performance depending on the within-context paragraphs - although retrieval is strongly helped by the context routing, the generation benefit is less stable in the settings tested here. Another consideration is the requirement for chunks to have contextual metadata for this approach to work.

2. How does the required LoRA rank change in these three applications, as the knowledge base becomes more challenging?

In the investigated setup, for LoRA as parametric memory, a rank higher than 4 appears necessary in a non-semantic setting in order to enable internalization of new knowledge when the number of new facts is in the realm of a few thousand. Moreover, the benefit of moving from rank 4 to rank 8 or higher is substantial, further cementing that a rank of at least 8 is preferred. In a semantic setting, no lowest rank to avoid parameter saturation from fact load is observed in the data regime used and the system setup tested, and performance increases more uniformly with rank than in the non-semantic setting. However, regardless of semantic or non-semantic setting, a rank of at least 8 is here necessary to reliably achieve answer correctness of 50% (at least for the larger dataset sizes). The study's findings also imply that LoRA learns different levels of the learning task for different ranks - for example, a lower rank is required to learn the expected format than learning the answer space, which in turn requires lower rank than learning the relationship between entities - indicating that the required rank changes with the learning task complexity. Another general finding is that learning semantic knowledge decreases the required rank as opposed to symbolic knowledge.

For LoRA in retrieval adaptation, benefits from increased rank appear for data sizes in the realm of a few thousand in the investigated setup, where increasing rank has a steady positive effect on both retrieval and generation performance. To achieve visible improvements over the RAG baseline for answer generation, a rank of 1 suffices, whereas increasing the rank leads to a monotonic and non-trivial increase in performance.

For LoRA in context routing, rank requirements depend on the classification difficulty and as such, the routing level. In the tested coarse routing levels (such as Company+Year in this thesis' setting), a rank of 1 sufficed in the setup used, because the route metadata was usually present in the test queries. However in settings with more granular and more difficult classification, such as where metadata is less explicit, rank requirements would likely increase substantially, although this isn't exhaustively tested in this thesis.

5.7 Future work

Future studies may expand on the findings of this report by investigating whether the same rank scaling patterns and system trade-offs appear in real-world datasets with similar structured properties. They may also study the effects of increasing the dataset sizes beyond what this study has done - this study stopped at 300 contexts, 1500 chunks and about 2500 facts, but moving into tens of thousands of facts would probably enable seeing clearer saturation effects of LoRA adapters in the semantic setting. In a similar vein, future work could consider investigating how base model size interacts with rank requirements for good knowledge internalization, and for rank requirements to avoid adapter saturation - it is probable that increasing base model size would decrease the rank needed, since the base model's weight matrix sizes dictate the total number of parameters being updated for a given LoRA rank.

It would also be interesting to more systematically compare rank requirements alongside training budget. This study has mainly been concerned with the performance under a fixed number of epochs, but lower ranks could possibly perform just as well as higher ranks at certain tasks if given enough training. The setup also meant more steps for larger datasets, and future studies could consider investigating the effects of a fixed number of training steps, or by using convergence-based stopping rules for training.

To further explore LoRA as a tool for embedding space modification, future work could explore refined training setups. For example, training could be done with more deliberately chosen hard negative chunks rather than uniformly sampled chunks, more closely resembling the retrieval task. It could also include a penalty for separating all chunks from each other, in order to encourage better clustering by pulling similar chunks closer together.

While LoRA has shown in this study to work well for routing in retrieval systems, the routing procedure could be refined. One possible direction could involve using confidence thresholds for routing to minimize incorrect routing, by only filtering if the router is confident in the prediction. Another direction is hierarchical routing, where the router first predicts the first context layer, then predicts the second layer conditional on both the query and the predicted first layer, and so on. Hierarchical routing could allow the system to decide filtering level based on how confident it is at

each context level. This could be combined with top-k routing to enable "pruning" the tree of contexts in a flexible way, where the depth and breadth of the context filtering depends on the confidence of the router.

The fact that improved retrieval doesn't automatically translate to improved generation, as made evident by the findings of System 4, could also inspire future work. Investigating how generation in RAG is affected by the composition of the retrieved chunks would shed light on how retrieval should best be improved for the best possible question-answering ability of the system, and it would help inform the usage of routing systems for retrieval.

A potentially interesting direction for future work is a more explicit study of the trade-off between robustness to different data regimes and computational efficiency across System 3 and System 4. In this study, the former appears more robust in converting routing performance into question-answering performance but is likely to become more computationally demanding to train as data size grows, whereas the latter shows the opposite characteristics. Whether these findings hold in more generalized settings remains an open direction to study. In a similar vein, it would be interesting to investigate how System 3 and System 4 behave as new potentially quite different data is added, to inspect their respective robustness to new data without retraining.

Another interesting prospect for future studies is the possibility of combining the three ways LoRA has been used in this study within one single system. Given for example a set training budget, such studies could investigate how to balance LoRA for parametric memory, embedding space adaptation, and routing within a single system, and how the optimal balance differs between use cases. For instance, coarse routing could restrict the search space after which an adapted embedding model could be applied within the selected contexts, and parametric memorization could be used on facts unlikely to change or on knowledge not requiring relational binding between entities (such as knowledge regarding expected format or even knowledge about the answer space). In all, a hybrid system like this could potentially capture the strengths of each approach while mitigating the weaknesses of each system tested in this thesis.

6
Conclusion

This thesis studied how LoRA can be used to improve question answering over structured document collections. To do this, three different roles of LoRA were investigated: as parametric memory, as a way to adapt the embedding model in a retrieval system, and as a routing mechanism to filter the search space before retrieval in a retrieval system. The experiments varied dataset sizes (N) and document collection confusability (S), and performed a rank sweep across a fixed set of ranks with a fixed number of epochs. The report focused on two main research areas: the benefits and characteristics of the three investigated LoRA roles, and the way rank requirements scale with size and difficulty across the roles.

The results show different advantages among the roles. LoRA as parametric memory gives strong question-answering improvement when increasing rank in the tested seen-fact/new-formulation setting, but it is also the least flexible of the systems when new knowledge is added. LoRA for embedding adaptation gives modest but robust improvements in both retrieval and generation, while LoRA for context routing gives strong retrieval gains with low rank requirements but less stable generation performance. Overall, the study argues that no single role seems universally superior, but rather that there is a trade-off between, for example, answer accuracy, flexibility, and robustness, all of which should be taken into account when designing a question-answering system.

For LoRA as parametric memory, the results indicate that rank requirements depend strongly on the type of knowledge the system is trained to memorize. In the semantic setting, performance improves more gradually with rank in a way that suggests that semantically meaningful content aids LoRA. In the non-semantic setting, where facts are opaque codes, a clearer rank threshold appears in the tested setting between ranks 4 and 8, in a way that indicates that arbitrary fact bindings place more pressure on LoRA capacity. The results further show that different levels of the learning task need different levels of LoRA capacity: lower ranks are enough to learn the expected answer format, higher ranks are required to memorize the answer space, and even higher ranks are needed to bind answers to contexts in a relational way that enables question-answering on specific facts.

For the RAG systems, the results show that RAG can be improved both by reshaping the embedding space with LoRA and by reducing the search space through context routing trained using LoRA. The former approach is shown to be a reliable improvement in the larger dataset regime, and provides a strong conversion

from retrieving the gold chunk to generating a correct answer. The latter approach is shown to give even stronger retrieval gains, and the results indicate that a semi-coarse filtering level was preferred in this setting to avoid filtering out the gold chunk. However, the routing approach also shows that filtering can have unexpected effects on generation, depending on the dataset regime.

The results also show that rank requirements are directly tied to the role LoRA fills. As parametric memory, where rank controls the capacity to internalize facts, LoRA is the most sensitive to rank. For embedding adaptation, low ranks suffice to improve the embedding space, with gradual improvements for higher ranks. For routing, a very low rank can work well, depending on how coarse the filtering is and how explicit metadata is in the queries used. Therefore, required rank is not only about dataset size and the setting's difficulty, but also about the type of task the LoRA adapter needs to learn.

The findings should be interpreted as specific to this controlled, synthetic context. The datasets make it possible to isolate the effects of dataset size, confusability, and LoRA rank, but they do not capture the full variability and noise of real-world document collections. With this in mind, the patterns found can inform design choices of systems making use of external knowledge, and may inspire future work regarding LoRA, retrieval, and structured document collections. Overall, the thesis suggests that LoRA should not be seen only as a fine-tuning method for storing knowledge with generally applicable characteristics. Rather, it is a flexible tool that can be utilized in different roles in question-answering systems, and its required rank depends strongly on the task it is used for: storing facts, adapting an embedding space, or fine-tuning a routing classifier.

Thesis Usage of AI Tools

To facilitate the coding of each system, including data handling and system logic, as well as the production of plots, the AI coding agents Codex by OpenAI and Claude Code by Anthropic were used. All elements of experimental design, system design, and analysis were performed by the authors.

Furthermore, AI language models were used to assist with improving the grammar and wording clarity of the report, though all full formulations are the authors' own.

Bibliography

[1] Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *arXiv preprint arXiv:2012.13255*, 2020.

[2] Anonymous. Post-hoc compression of lora adapters via singular value decomposition, 2026. Under review as a conference paper at ICLR 2026.

[3] Anthropic. Claude Code overview. `https://code.claude.com/docs/en/overview`, n.d. Claude Code documentation overview page; accessed 2026-05-08.

[4] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*, 2023.

[5] Yein Choi, Sungwoo Kim, Yipene Cedric Francois Bassole, and Yunsick Sung. Enhanced retrieval-augmented generation using low-rank adaptation. *Applied Sciences*, 15(8):4425, 2025.

[6] Cristian Leo. FAISS & RAG: The Dynamic Duo of Knowledge-Powered AI. `https://levelup.gitconnected.com/faiss-rag-the-dynamic-duo-of-knowledge-powered-ai-7d6bc2ced781`, 2025. Level Up Coding article; accessed 2026-05-05.

[7] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient fine-tuning of quantized large language models. *arXiv preprint arXiv:2305.14314*, 2023.

[8] Ning Ding, Yujia Qin, Guohong Du, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 2023.

[9] Feng Wang and Huaping Liu. Understanding the Behaviour of Contrastive Loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2495–2504, June 2021.

[10] Florin Cuconasu, Giovanni Trappolini, Nicola Tonellotto, and Fabrizio Silvestri. A Tale of Trust and Accuracy: Base vs. Instruct LLMs in RAG Systems. `https://arxiv.org/pdf/2406.14972`, 2024. arXiv preprint arXiv:2406.14972; accessed 2026-05-06.

[11] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

[12] GeeksforGeeks. Evaluation Metrics for Retrieval-Augmented Generation (RAG) Systems. `https://www.geeksforgeeks.org/nlp/evaluation-metrics-for-retrieval-augmented-generation-rag-systems/`, 2025. GeeksforGeeks article; last updated 2025-12-16; accessed 2026-05-08.

[13] Hao Yu and Aoran Gan and Kai Zhang and Shiwei Tong and Qi Liu and Zhaofeng Liu. Evaluation of Retrieval-Augmented Generation: A Survey. https://arxiv.org/html/2405.07437v2, 2024. arXiv preprint arXiv:2405.07437v2; last revised 2024-07-03; accessed 2026-05-08.

[14] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[15] Hugging Face. LoRA (Low-Rank Adaptation). https://huggingface.co/learn/llm-course/en/chapter11/4, n.d. Hugging Face LLM Course documentation; accessed 2026-05-05.

[16] Ishan Jindal, Chandana Badrinath, Pranjal Bharti, Lakkidi Vinay, and Sachin Dev Sharma. Balancing Continuous Pre-Training and Instruction Fine-Tuning: Optimizing Instruction-Following in LLMs. https://arxiv.org/html/2410.10739v1, 2024. arXiv preprint arXiv:2410.10739v1; accessed 2026-05-06.

[17] Jagadeesan Ganesh. Mastering Chunking in Retrieval-Augmented Generation (RAG): 6 Powerful Techniques with Examples. https://medium.com/@jagadeesan.ganesh/mastering-chunking-in-retrieval-augmented-generation-rag-6-powerful-techniques-with-examples-767db2deb9a3, 2024. Medium article; accessed 2026-05-08.

[18] Jon Gitlin and David Dalmaso. Why normalized data is critical for best-in-class retrieval-augmented generation (RAG). https://www.merge.dev/blog/normalized-data-rag, n.d. Merge blog post; accessed 2026-05-05.

[19] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. 2023. 3rd edition draft.

[20] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical Review E*, 69(6):066138, 2004.

[21] Lalatendu Keshari Swain. Choosing the Right Small Language Model for RAG: A Comprehensive Comparison Guide. https://lalatenduswain.medium.com/choosing-the-right-small-language-model-for-rag-a-comprehensive-comparison-guide-6e60044441ac, 2026. Medium article; accessed 2026-05-06.

[22] LangChain. Build a RAG agent with LangChain. https://docs.langchain.com/oss/python/langchain/rag, n.d. LangChain documentation; accessed 2026-05-06.

[23] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2005.11401*, 2020. Accepted at NeurIPS 2020.

[24] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, 2008.

[25] Muazma Zahid. Similarity Search with FAISS and Azure SQL. https://devblogs.microsoft.com/azure-sql/faiss-and-azure-sql/, 2024. Microsoft Azure SQL Dev Corner blog post; accessed 2026-05-05.

[26] Agada Joseph Oche, Ademola Glory Folashade, Tirthankar Ghosal, and Arpan Biswas. A systematic review of key retrieval-augmented generation (rag) sys-

tems: Progress, gaps, and future directions. *arXiv preprint arXiv:2507.18910*, 2025.

[27] OpenAI. Codex | AI-kodningspartner från OpenAI. `https://openai.com/sv-SE/codex/`, n.d. OpenAI product page; accessed 2026-05-08.

[28] Yuefeng Peng, Junda Wang, Hong Yu, and Amir Houmansadr. Data extraction attacks in retrieval-augmented generation via backdoors. *arXiv preprint arXiv:2411.01705v2*, 2025. 30 Mar 2025.

[29] Pengyue Hou and Xingyu Li. Improving Contrastive Learning of Sentence Embeddings with Focal-InfoNCE. `https://arxiv.org/html/2310.06918v1`, 2023. arXiv preprint arXiv:2310.06918v1; accessed 2026-05-06.

[30] Pere Martra. Implementing semantic cache to improve a RAG system with FAISS. `https://huggingface.co/learn/cookbook/semantic_cache_chroma_vector_database`, n.d. Hugging Face Open-Source AI Cookbook; accessed 2026-05-05.

[31] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*, 2019.

[32] Sergey Pletenev, Maria Marina, Daniil Moskovskiy, Vasily Konovalov, Pavel Braslavski, Alexander Panchenko, and Mikhail Salnikov. How much knowledge can you pack into a lora adapter without harming llm? *Findings of NAACL 2025*, 2025. arXiv:2502.14502.

[33] Priya Kulkarni. Vector Databases for RAG: FAISS vs Chroma vs Pinecone. `https://medium.com/@priyaskulkarni/vector-databases-for-rag-faiss-vs-chroma-vs-pinecone-6797bd98277d`, 2025. Medium article; accessed 2026-05-05.

[34] Qwen Team. Qwen2.5-0.5B. `https://huggingface.co/Qwen/Qwen2.5-0.5B`, 2024. Hugging Face model card; accessed 2026-05-05.

[35] Qwen Team. Qwen2.5-0.5B-Instruct. `https://huggingface.co/Qwen/Qwen2.5-0.5B-Instruct`, 2024. Hugging Face model card; accessed 2026-05-06.

[36] Qwen Team. Qwen3-Embedding-0.6B. `https://huggingface.co/Qwen/Qwen3-Embedding-0.6B`, 2025. Hugging Face model card; accessed 2026-05-05.

[37] Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*, 2020.

[38] Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *Proceedings of the 15th European Signal Processing Conference (EUSIPCO)*, pages 606–610, Poznan, Poland, 2007.

[39] scikit-learn developers. sklearn.feature_selection.mutual_info_regression. `https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_regression.html#id5`, 2026. scikit-learn 1.8.0 documentation; accessed 2026-05-03.

[40] Sentence Transformers developers. SentenceTransformer. `https://sbert.net/docs/package_reference/sentence_transformer/model.html`, 2026. Sentence Transformers documentation; accessed 2026-05-05.

[41] Sergio Perez Rodriguez. Vector Databases for Efficient RAG: A First Look at FAISS and Redis. `https://medium.com/@sergiopr89/vector-databases`

-for-efficient-rag-a-first-look-at-faiss-and-redis-b18314cd30b6, 2025. Medium article; accessed 2026-05-05.

[42] Sharon Campbell-Crow. Retrieval-Augmented Generation: A Practical Guide to RAG Architecture, Retrieval, and Production-Ready Context. https://www.comet.com/site/blog/retrieval-augmented-generation/, 2026. Comet blog post; accessed 2026-05-05.

[43] Yixuan Su, Reza Mirafzal, and Julie Stal-Le Cardinal. Exploring llm-based agents for need analysis of knowledge management practice. *Proceedings of the Design Society*, 5:1675–1684, 2025.

[44] Zhan Su, Fengran Mo, Jinghan Zhang, Yuchen Hui, Jiaao Sun, and Jian-yun Nie. Parametric retrieval-augmented generation using latent routing of lora adapters. *arXiv preprint arXiv:2511.17044*, 2025.

[45] Tayyib Ul Hassan Gondal. One stop guide for QLoRA. https://medium.com/@tayyibgondal2003/one-stop-guide-for-qlora-72abbad9fd0f, 2024. Medium article; accessed 2026-05-05.

[46] The SciPy Community. scipy.stats.pearsonr. https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.pearsonr.html, 2026. SciPy v1.17.0 Manual; accessed 2026-05-03.

[47] Unsloth AI. LoRA fine-tuning Hyperparameters Guide. https://unsloth.ai/docs/get-started/fine-tuning-llms-guide/lora-hyperparameters-guide, 2026. Unsloth documentation; accessed 2026-05-05.

[48] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*, 2020.

[49] Younes B, Tim Dettmers, Artidoro Pagnoni, Sylvain Gugger, and Sourab Mangrulkar. Making LLMs even more accessible with bitsandbytes, 4-bit quantization and QLoRA. https://huggingface.co/blog/4bit-transformers-bitsandbytes, 2023. Hugging Face blog post; accessed 2026-05-05.

A

Appendix 1 - Method

A.1 System specific settings

Table A.1: System 1 settings.

Component	Setting	Explanation	Motivation
Base model	Qwen2.5-0.5B	Model LoRA is applied to	Common, small base LLM [34]
Training method	QLoRA	PEFT method	Common LoRA variant using quantized model to decrease memory load [7, 49]
Quantization	NF4, double quantization enabled, bf16 compute dtype where supported	Reduces memory use during training	Common settings [49]
LoRA ranks	$r \in \{1, 2, 4, 8, 16\}$	Adapter capacity	Choice inspired by original LoRA paper, range modified for experimental interpretability [14]
LoRA alpha	$\alpha = 2r$	Scales the LoRA update	Common setting [15]
LoRA target modules	`q_proj`, `k_proj`, `v_proj`, `o_proj`	Attention projections where LoRA is applied	Based on original LoRA paper [14]
LoRA dropout	0.05	Regularizer during LoRA training	Based on original QLoRA paper [7]
Batch configuration	Per-device batch size = 2, gradient accumulation = 8, gives effective batch size = 16	Number of training examples used per update step	Common setting [47]
Optimizer	Paged AdamW 8-bit (`paged_adamw_8bit`)	Optimization algorithm	Based on original QLoRA paper and common recommendations [7, 45]
Learning rate	2×10^{-4}	Optimization step size	Common recommendation [47]
Weight decay	0.01	Regularizes weight sizes	Common recommendation [47]
Scheduler/warmup	Cosine scheduler, 5% warmup	Controls learning rate change during training	Common recommendation [47]
Gradient clipping	Max gradient norm = 0.3	Limit to update size	Based on recommendations [45]
Standard epochs	24	Number of passes through the training data per individual run	Design choice
Seeds	3 seeds per configuration	Number of runs per configuration	Design choice based on the project-specific trade-off between reliability of conclusions and computational feasibility

Table A.2: System 2 settings.

Component	Setting	Explanation	Motivation
Embedding model	`Qwen/Qwen3-Embedding-0.6B`	Model used to embed chunks and queries into vectors	Common, small embedding model [36]
Embedding normalization	Enabled	Scales embeddings to the same length	Based on recommendations, and needed for the chosen FAISS retriever index to use cosine similarity as similarity metric, since cosine similarity equals the normalized dot product [18, 25]
Document/chunk encoding	No prompt prefix	Additional prefixes added to chunks before embedding	Chosen to omit for simplicity
Query encoding	`prompt_name="query"`	Encoding used to enable embedding model to treat queries correctly	Choice inspired by documentation [40]
Embedding dimension	1024	Size of each embedding vector produced by the embedding model	Largest supported dimensionality for chosen embedding model [36]
Retriever index	FAISS `IndexFlatIP`	Database of embeddings	Popular choice for RAG systems, and `IndexFlatIP` uses dot product. Combined with normalization of embeddings, this gives cosine similarity [6, 41, 33, 30, 25]
Index batch size	64	Number of chunks processed simultaneously when building index	Defaults to 32, choice to double it to improve index building speed [40]
Chunking method	Paragraph chunking	The approach to splitting documents into chunks	One of the common chunking methods, design choice allowing for controlled chunking and clear control in the project's synthetic data regime [17]
Retrieved chunks	$k = 5$	Number of top-ranked chunks retrieved as context for the generator	Middle-ground of common values [42]
Generator	`Qwen/Qwen2.5-0.5B-Instruct`	LLM used to generate the answer	Common, small base model. The instruct version of the same base model fine-tuned using LoRA in System 1 was chosen to make use of the retrieved contexts and avoid guessing. Some recommend using instruct versions for RAG and base versions for fine-tuning, though the topic is complex and context-dependent [21, 10, 16]
Generation	`max_new_tokens` = 128, temperature = 0.0, `do_sample=false`	Settings for answer length and randomness	Deterministic generation chosen for reproducibility, and long enough generation limit that the model may answer correctly to any query in the dataset
Max input tokens	2048	Limit to the amount of input text the generator can use	Limit chosen to leave headroom for any experiment-specific input possible, but not high just for the sake of it
Max context length	4000 characters	Limit to the amount of retrieved chunk text the generator can use	Large enough to allow any retrieved context set combination, since 5 chunks are retrieved, but still fit within the model [35]

Table A.2 continued.

Component	Setting	Explanation	Motivation
Prompt rule	Answer only using retrieved context; abstain if unsupported	Instruction on answer policy	Inspired by recommendations [22]

Table A.3: System 3 settings.

Component	Setting	Explanation	Motivation
Embedding model	`Qwen/Qwen3-Embedding-0.6B`	Model used to embed chunks and queries into vectors	Common, small embedding model [36]
Embedding normalization	Enabled	Scales embeddings to the same length	Based on recommendations, and needed for the chosen FAISS retriever index to use cosine similarity as similarity metric, since cosine similarity equals the normalized dot product [18, 25]
Document/chunk encoding	No prompt prefix	Additional prefixes added to chunks before embedding	Chosen to omit for simplicity
Query encoding	`prompt_name="query"`	Encoding used to enable embedding model to treat queries correctly	Choice inspired by documentation [40]
Negatives per query	4	Number of incorrect chunks exposed to the model per query during training	Design choice for training
Contrastive temperature	0.07	Controls strength of separation between gold chunk and negatives during training	Commonly used, medium-sized setting for contrastive temperature [9, 29]
Training method	QLoRA	PEFT method	Common LoRA variant using quantized model to decrease memory load [7, 49]
Quantization	NF4, double quantization enabled, bf16 compute dtype where supported	Reduces memory use during training	Common settings [49]
LoRA ranks	$r \in \{1, 2, 4, 8, 16\}$	Adapter capacity	Choice inspired by original LoRA paper, range modified for experimental interpretability [14]
LoRA alpha	$\alpha = 2r$	Scales the LoRA update	Common setting [15]
LoRA target modules	`q_proj`, `k_proj`, `v_proj`, `o_proj`	Attention projections where LoRA is applied	Based on original LoRA paper [14]
LoRA dropout	0.05	Regularizer during LoRA training	Based on original QLoRA paper [7]
Training batch configuration	Batch size = 16	Number of training examples used per update step	Common setting [47]
Optimizer	`PagedAdamW32bit`	Optimization algorithm	Based on original QLoRA paper and common recommendations [7, 45]
Learning rate	2×10^{-4}	Optimization step size	Common recommendation [47]
Weight decay	0.01	Regularizes weight sizes	Common recommendation [47]
Scheduler/warmup	Cosine scheduler, 5% warmup	Controls learning rate change during training	Common recommendation [47]
Gradient clipping	Max gradient norm = 0.3	Limit to update size	Based on recommendations [45]
Standard epochs	24	Number of passes through the training data per individual run	Design choice

Table A.3 continued.

Component	Setting	Explanation	Motivation
Seeds	3 seeds per configuration	Number of runs per configuration	Design choice based on the project-specific trade-off between reliability of conclusions and computational feasibility
Embedding dimension	1024	Size of each embedding vector produced by the embedding model	Largest supported dimensionality for chosen embedding model [36]
Retriever index	FAISS `IndexFlatIP`	Database of embeddings	Popular choice for RAG systems, and `IndexFlatIP` uses dot product. Combined with normalization of embeddings, this gives cosine similarity [6, 41, 33, 30, 25]
Index batch size	64	Number of chunks processed simultaneously when building index	Defaults to 32, choice to double it to improve index building speed [40]
Chunking method	Paragraph chunking	The approach to splitting documents into chunks	One of the common chunking methods, design choice allowing for controlled chunking and clear control in the project's synthetic data regime [17]
Retrieved chunks	$k = 5$	Number of top-ranked chunks retrieved as context for the generator	Middle-ground of common values [42]
Generator	`Qwen/Qwen2.5-0.5B-Instruct`	LLM used to generate the answer	Common, small base model. The instruct version of the same base model fine-tuned using LoRA in System 1 was chosen to make use of the retrieved contexts and avoid guessing. Some recommend using instruct versions for RAG and base versions for fine-tuning, though the topic is complex and context-dependent [21, 10, 16]
Generation	`max_new_tokens` = 128, temperature = 0.0, `do_sample=false`	Settings for answer length and randomness	Deterministic generation chosen for reproducibility, and long enough generation limit that the model may answer correctly to any query in the dataset
Max input tokens	2048	Limit to the amount of input text the generator can use	Limit chosen to leave headroom for any experiment-specific input possible, but not high just for the sake of it
Max context length	4000 characters	Limit to the amount of retrieved chunk text the generator can use	Large enough to allow any retrieved context set combination, since 5 chunks are retrieved, but still fit within the model [35]
Prompt rule	Answer only using retrieved context; abstain if unsupported	Instruction on answer policy	Inspired by recommendations [22]

Table A.4: System 4 settings.

Component	Setting	Explanation	Motivation
Router base model	`Qwen/Qwen2.5-0.5B` with sequence-classification head	Base model to which LoRA is applied for route classification	Common, small base LLM [34]
Training method	QLoRA	PEFT method applied to the router only	Common LoRA variant using quantized model to decrease memory load [7, 49]
Quantization	NF4, double quantization enabled, bf16 compute dtype where supported	Reduces memory use during training	Common settings [49]
Routing levels	Company, Company+Year, Company+Year+DocumentType, Company+Year-DocumentType+Section	Filter levels tried, from coarse to fine	Subset of all possible contextual filters, chosen as a trade-off between experimental interpretability and computational feasibility (all possible filter combinations would be excessive)
Filter policy	Strict filtering; top-2 and top-3 route candidates used as diagnostic variants	Whether to filter on the predicted route class or on a larger set of likely classes	Experimental design choice
Fallback policy	Fall back to global retrieval if filtered subset is empty	Handling of predicted routes with no chunks	Experimental design choice for robustness
LoRA ranks	$r \in \{1, 2, 4, 8, 16\}$ for Company+Year routing, 16 for the remaining	Adapter capacity	Choice inspired by original LoRA paper, range modified for experimental interpretability [14]
LoRA alpha	$\alpha = 2r$	Scales the LoRA update	Common setting [15]
LoRA target modules	`q_proj`, `k_proj`, `v_proj`, `o_proj`	Attention projections where LoRA is applied	Based on original LoRA paper [14]
LoRA dropout	0.05	Regularizer during LoRA training	Based on original QLoRA paper [7]
Training batch configuration	Batch size = 16	Number of training examples used per update step	Common setting [47]
Optimizer	`PagedAdamW32bit`	Optimization algorithm	Based on original QLoRA paper and common recommendations [7, 45]
Learning rate	2×10^{-4}	Optimization step size	Common recommendation [47]
Weight decay	0.01	Regularizes weight sizes	Common recommendation [47]
Scheduler/warmup	Cosine scheduler, 5% warmup	Controls learning rate change during training	Common recommendation [47]
Gradient clipping	Max gradient norm = 0.3	Limit to update size	Based on recommendations [45]
Standard epochs	24	Number of passes through the router training data per individual run	Design choice
Router max input length	256 tokens	Maximum tokenized input length for the router model	Covers any query in this experimental setup
Seeds	3 seeds per configuration	Number of runs per configuration	Design choice based on the project-specific trade-off between reliability of conclusions and computational feasibility
Embedding model	`Qwen/Qwen3-Embedding-0.6B`	Model used to embed chunks and queries into vectors	Common, small embedding model [36]

Table A.4 continued.

Component	Setting	Explanation	Motivation
Embedding normalization	Enabled	Scales embeddings to the same length	Based on recommendations, and needed for the chosen FAISS retriever index to use cosine similarity as similarity metric, since cosine similarity equals the normalized dot product [18, 25]
Document/chunk encoding	No prompt prefix	Additional prefixes added to chunks before embedding	Chosen to omit for simplicity
Query encoding	`prompt_name="query"`	Encoding used to enable embedding model to treat queries correctly	Choice inspired by documentation [40]
Embedding dimension	1024	Size of each embedding vector produced by the embedding model	Largest supported dimensionality for chosen embedding model [36]
Retriever index	FAISS `IndexFlatIP`	Database of embeddings	Popular choice for RAG systems, and `IndexFlatIP` uses dot product. Combined with normalization of embeddings, this gives cosine similarity [6, 41, 33, 30, 25]
Index batch size	64	Number of chunks processed simultaneously when building index	Defaults to 32, choice to double it to improve index building speed [40]
Chunking method	Paragraph chunking	The approach to splitting documents into chunks	One of the common chunking methods, design choice allowing for controlled chunking and clear control in the project's synthetic data regime [17]
Retrieved chunks	$k = 5$	Number of top-ranked chunks retrieved as context for the generator	Middle-ground of common values [42]
Generator	`Qwen/Qwen2.5-0.5B-Instruct`	LLM used to generate the answer	Common, small base model. The instruct version of the same base model fine-tuned using LoRA in System 1 was chosen to make use of the retrieved contexts and avoid guessing. Some recommend using instruct versions for RAG and base versions for fine-tuning, though the topic is complex and context-dependent [21, 10, 16]
Generation	`max_new_tokens` = 128, temperature = 0.0, `do_sample=false`	Settings for answer length and randomness	Deterministic generation chosen for reproducibility, and long enough generation limit that the model may answer correctly to any query in the dataset
Max input tokens	2048	Limit to the amount of input text the generator can use	Limit chosen to leave headroom for any experiment-specific input possible, but not high just for the sake of it
Max context length	4000 characters	Limit to the amount of retrieved chunk text the generator can use	Large enough to allow any retrieved context set combination, since 5 chunks are retrieved, but still fit within the model [35]
Prompt rule	Answer only using retrieved context; abstain if unsupported	Instruction on answer policy	Inspired by recommendations [22]

A.2 Experimental runs

Table A.5 summarizes the experimental training runs and evaluation-only runs, whose results are reported in this thesis. Note that System 1 uses the original N300 (semantic dataset), whereas Systems 2-4 use the deduplicated N300 versions, where accidental repetition of facts between chunks has been removed. This was done, for example, to avoid a confound where retrieving a correct chunk could be labeled incorrect in evaluation, and to generally ensure fair and interpretable evaluation. This resulted in all systems being tested on the same facts in the same test queries during evaluation.

Table A.5: Experimental trained runs and evaluation-only runs used for the results presented in the report.

Run family / diagnostic	**Data**	**N values**	**S values**	**Rank(s)**	**Seeds / source runs**
Trained runs					
System 1 semantic	Shared datasets	12, 30, 90, 300	S_{low}, S_{high}	1, 2, 4, 8, 16	3
System 1 stress	Stress data	12, 30, 90, 300	–	1, 2, 4, 8, 16	3
System 2	Shared datasets	12, 30, 90, 300	S_{low}, S_{high}	–	1
System 3	Shared datasets	12, 30, 90, 300	S_{low}, S_{high}	1, 2, 4, 8, 16	3
System 4 routing-level sweep	Shared datasets	12, 30, 90, 300	S_{low}, S_{high}	16	3
System 4 CompanyYear rank sweep	Shared datasets	12, 30, 90, 300	S_{low}, S_{high}	1, 2, 4, 8, 16	3
Evaluation-only runs					
System 4 oracle routing	Shared datasets	300	S_{low}, S_{high}	–	Gold-route evaluation with no trained router
System 4 top-k routing	Shared datasets	300	S_{low}, S_{high}	16	Reuses 3 trained r16 router seeds; evaluates top-1, top-2, and top-3 route candidates

B

Appendix 2 - Results

B.1 System 1

In the following section, additional figures presenting the results from System 1 are shown.

B.1.1 Semantic data

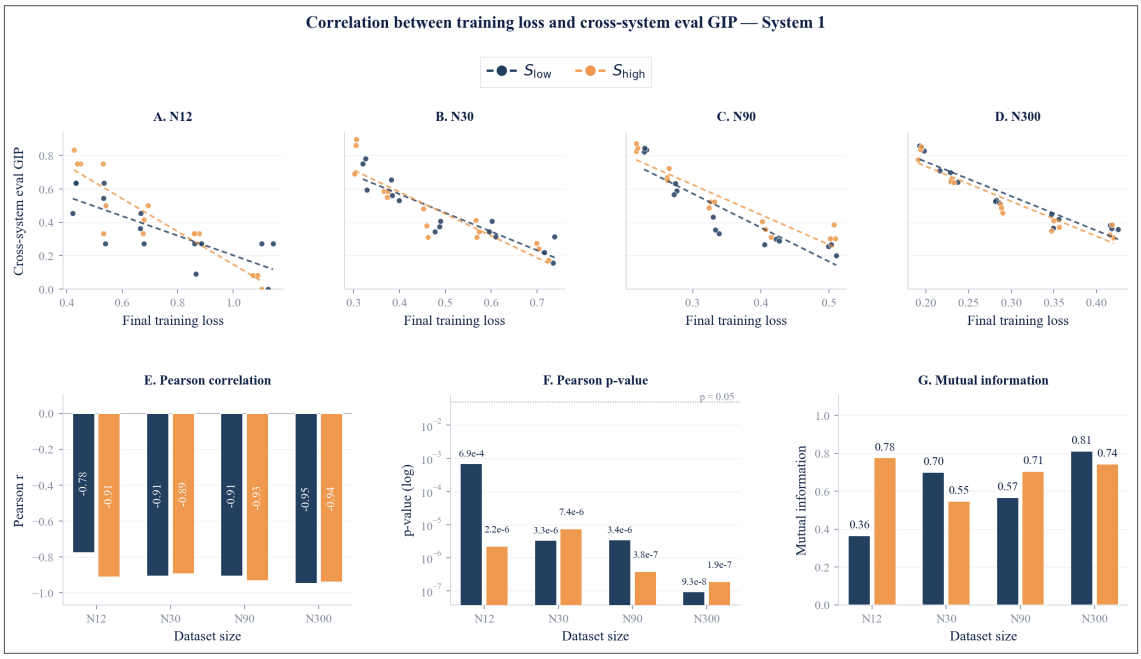

Figure B.1: Correlation between training loss and performance for System 1 in the semantic setting.

Figure B.1 shows the correlation between training loss and GIP-performance. In the scatter plots, each dot is an individual run. As can be seen, the Pearson r value is close to -1, with an associated p-value well below the 5% confidence threshold. Together, this shows a strong, negative relationship between the training task and the evaluation metric, which we can be very confident in. This is further supported by the Mutual Information, also showing strong signal.

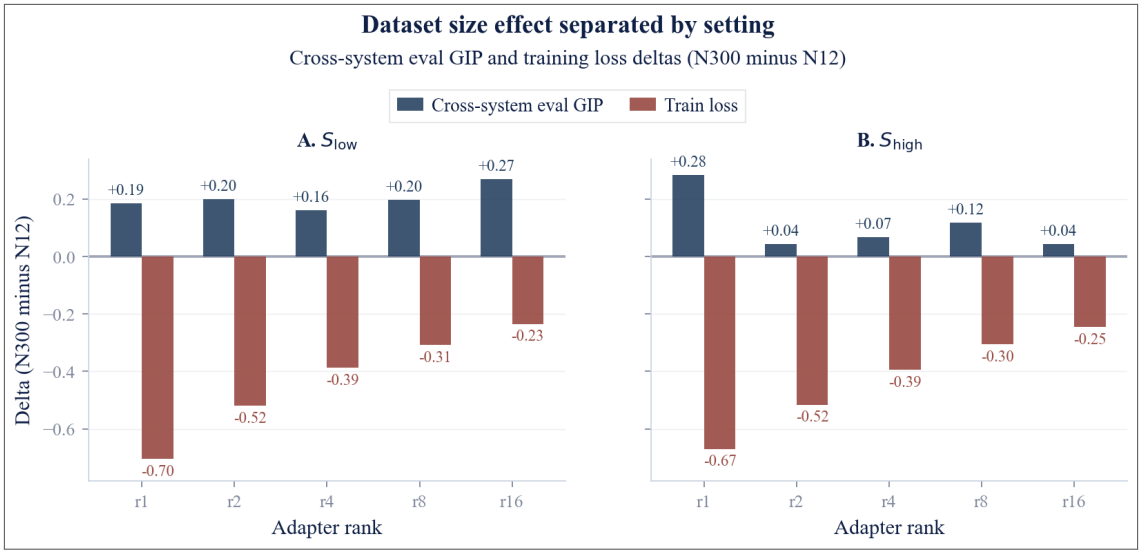

Figure B.2: Delta between N300 and N12 for System 1 in the semantic setting.

Figure B.2 shows the delta between N300 and N12 train loss and GIP accuracy across ranks. The red bars show that the loss for N12 datasets is always larger than for N300, a difference that decreases with increased rank. The blue bars show that GIP is constantly higher for N300 than for N12, and the difference doesn't change much as rank increases. Moreover, the improvements from N12 to N300 in terms of accuracy, for larger ranks, are more prominent for the S_{low} data than for S_{high}.

B.1.2 Stress data

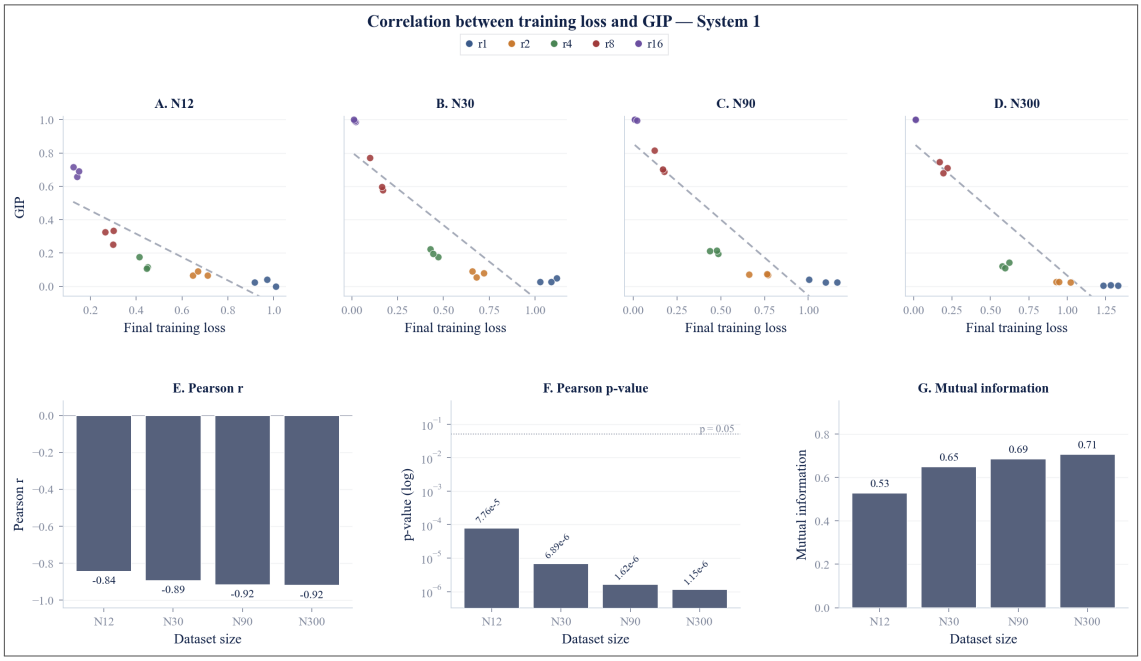

Figure B.3: Correlation between training loss and performance for System 1 in the stress setting.

Figure B.3 is the stress-equivalent of Figure B.1, showing the correlation between training loss and GIP-performance. There's a clear negative relationship between training loss and Pearson r, backed by low p values and high Mutual Information. The correlation is especially clear for larger dataset sizes. Compared to the semantic data, the correlation figures are not quite as strong, but still convincing.

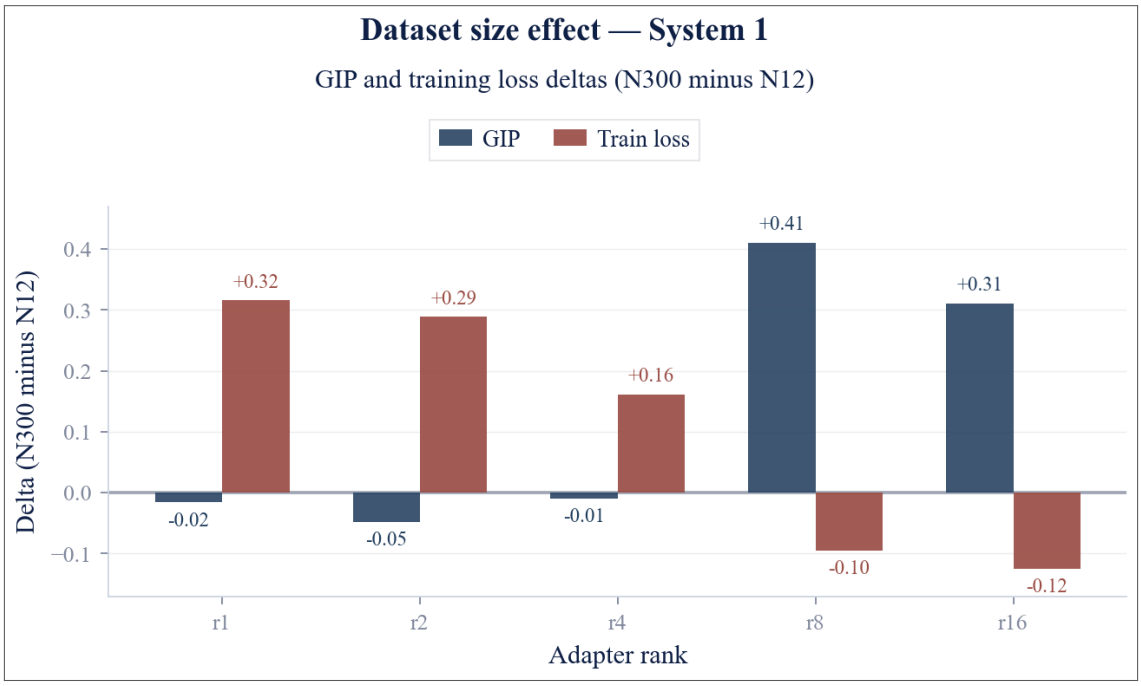

Figure B.4: Delta between N300 and N12 for System 1 in the stress setting.

Figure B.4 shows the delta between N300 and N12 train loss and GIP accuracy across ranks. The red bars show that for ranks 1, 2, and 4, the train loss is higher for N300 datasets than for N12, an effect which switches as rank increases further. The blue bars show that the accuracy is lower for N300 than for N12 datasets for the ranks 1, 2, and 4, but this switches quite drastically as rank increases to 8.

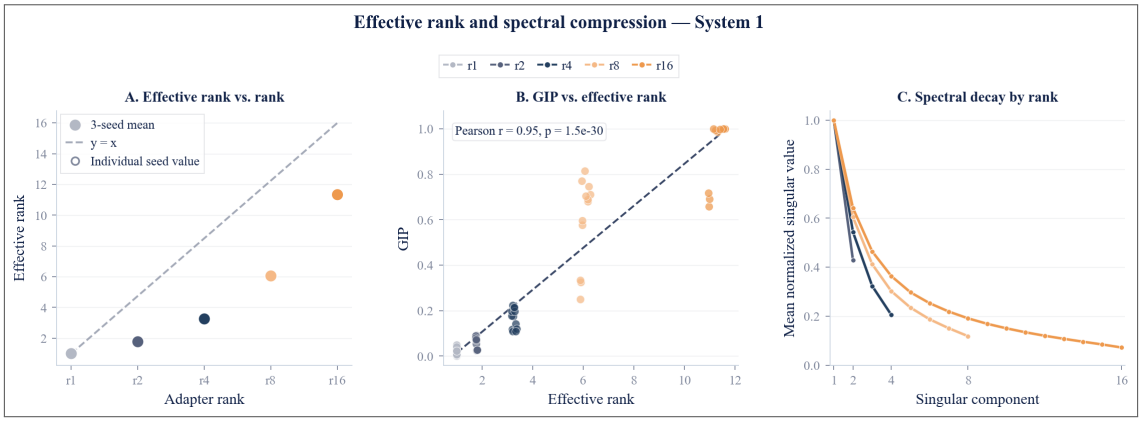

Figure B.5: Effective rank and spectral compression for System 1 in the stress setting.

Figure B.5 is the stress data equivalent of Figure 4.4 for the semantic data. Similar patterns are visible. The only difference that appears somewhat major is seen in

Panel B - the effect of effective rank on accuracy is more aggressive than for the semantic data, and the vertical spread is more prominent.

B.2 System 2

In the following section, additional figures presenting the results from System 2 are shown.

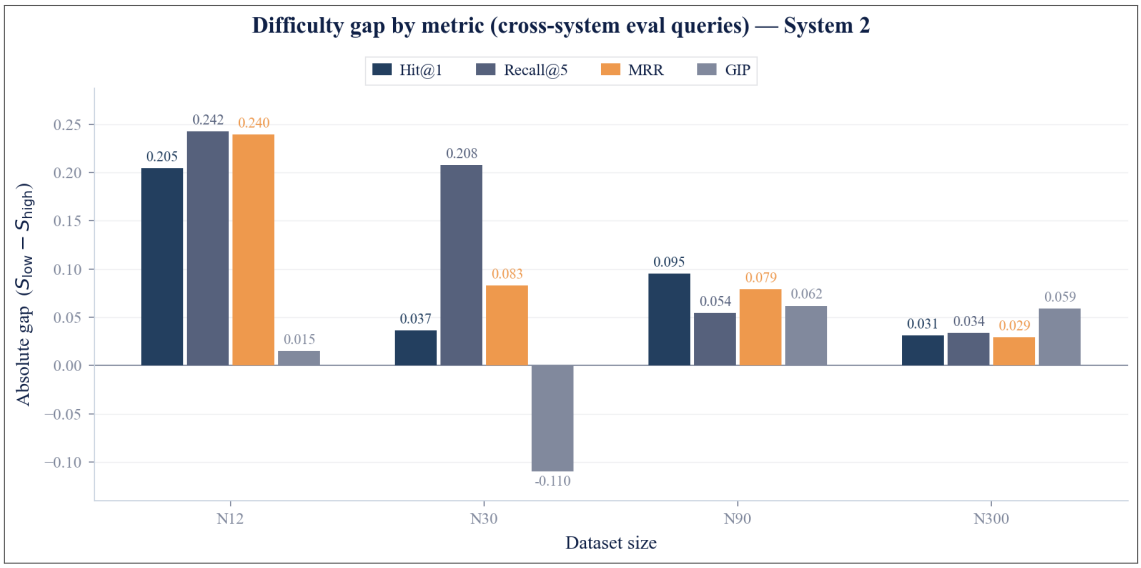

Figure B.6: Absolute performance gap between S_{low} and S_{high} for System 2 (S_{low} - S_{high}).

Figure B.6 shows the absolute gap between S_{low} and S_{high} for each metric. Each bar represents the difference between the two settings for a given metric and corpus size. For example, a value of 0.25 means that S_{low} performs 25 percentage points better than S_{high} on that metric. We can see that S_{low} results in higher scores than S_{high} across most metrics and dataset sizes, and that the gap in the performance metrics is generally larger for smaller datasets than for the larger datasets.

B.3 System 3

In the following section, additional figures presenting the results from System 3 are shown.

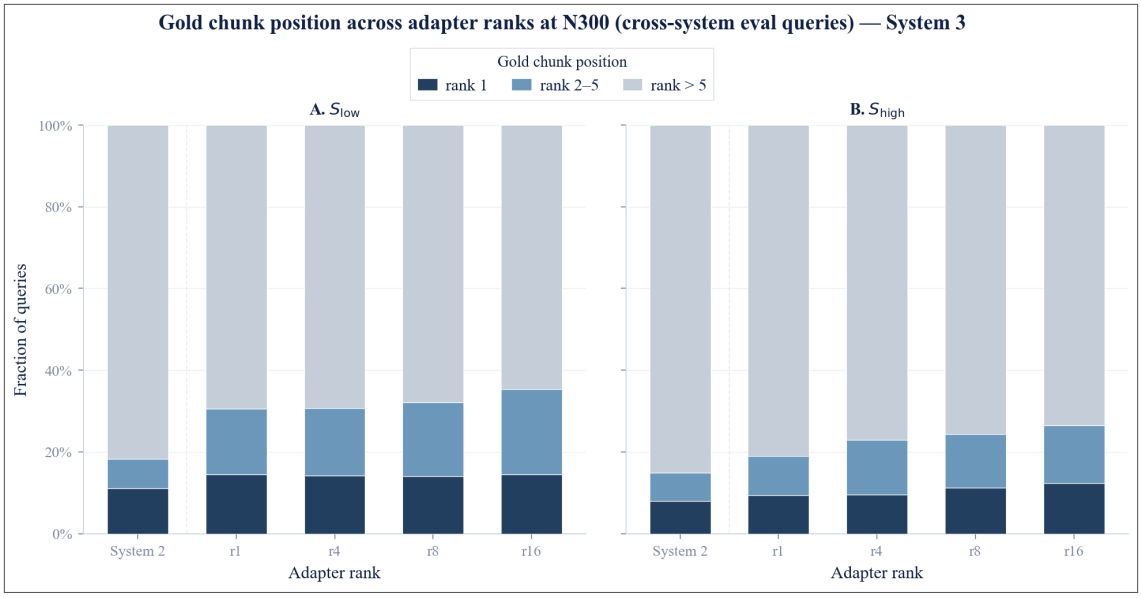

Figure B.7: Gold rank placement across ranks for System 3 (N300).

Figure B.7 shows how the gold chunk ranking changes with LoRA training, and as LoRA rank increases. Overall, increasing rank appears to increase the ranking of the gold chunk.

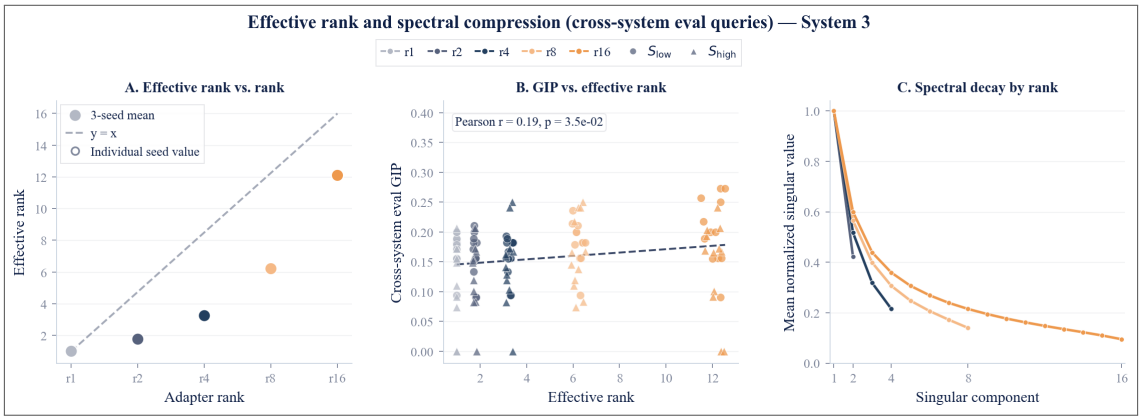

Figure B.8: Effective rank and spectral compression for System 3.

Figure B.8 shows an analysis of the effective rank as the rank hyperparameter increases, similarly as described previously for System 1 (see Figure 4.4). Similarly to the results seen in System 1, effective rank stays lower than the rank value chosen, there's a clear correlation between effective rank and Recall@5, and as Panel C shows how, as rank increases, the utilization of the available increase in dimensions decreases.

B.4 System 4

In the following section, additional figures presenting the results from System 4 are shown.

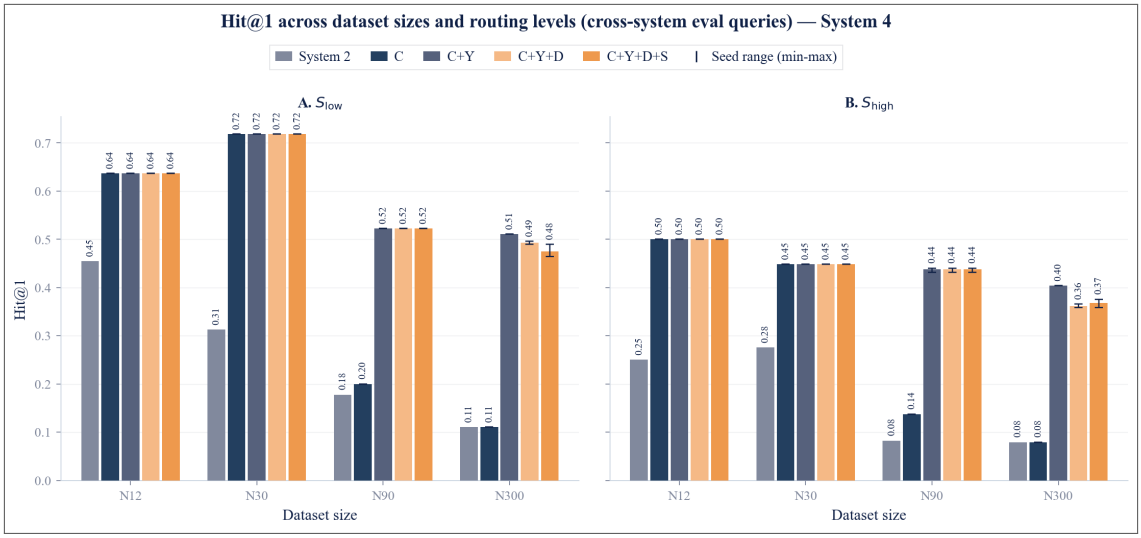

Figure B.9: System 4 Hit@1 across dataset sizes and routing levels (r=16).

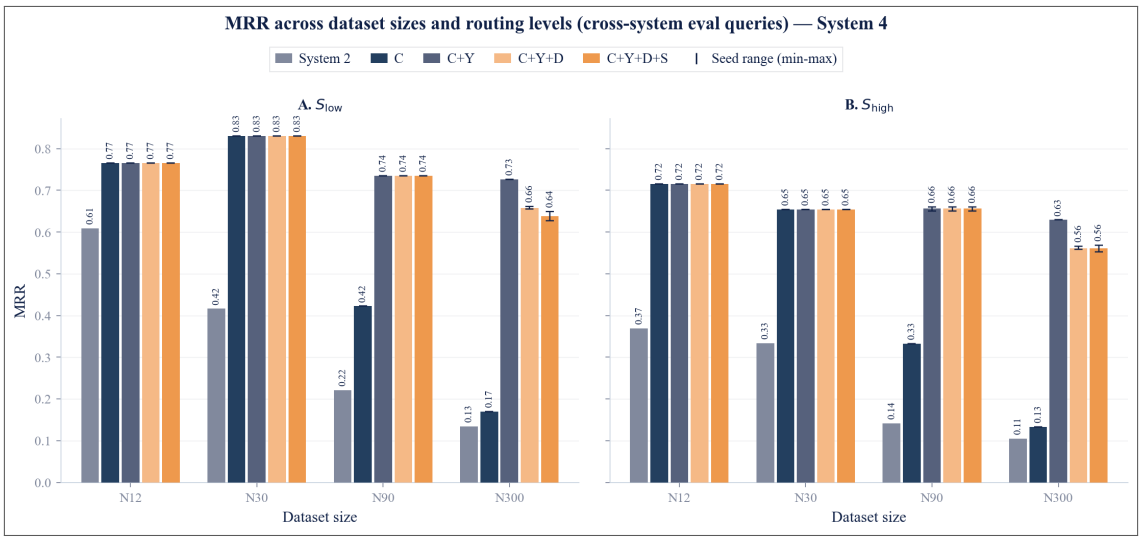

Figure B.10: System 4 MRR across dataset sizes and routing levels (r=16).

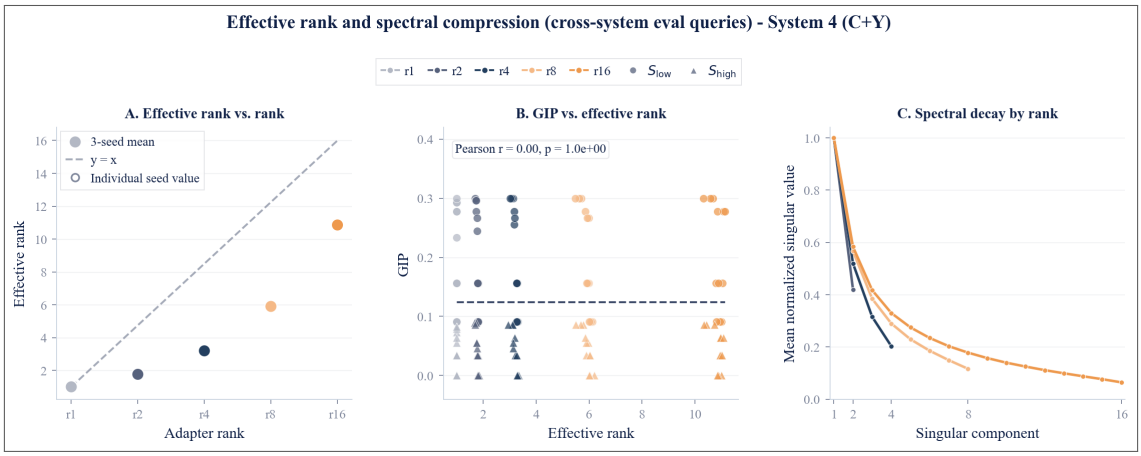

Figure B.11: Effective rank and spectral compression for System 4.

Figure B.11 shows an analysis of the effective rank as the rank hyperparameter increases, similarly as described previously for System 1 and System 3, with similar results. Panel B shows the largest difference to the other systems, with almost no correlation between GIP and effective rank.

Why System 4 struggles on S_{high} despite good retrieval

Example 1: N300 S_{high}

Query: What market conditions were reported for Solvane Logistics in 2022 in Middle East?
Gold answer: `stable in service-led demand`

System 3 output: The market conditions for Solvane Logistics in 2022 in Middle East were stable in service-led demand, led by a core segment.
System 4 output: There is no mention of market conditions for Solvane Logistics in the Middle East in 2022. The correct response would be: "I don't know."

System 3 retrieved chunks

1. **GOLD.** Market conditions for Solvane Logistics were stable in service-led demand, led by a core segment. Against that backdrop, a major risk exposure remained a material risk item for Solvane Logistics, prompting a mitigation program.
2. Sales for the company expanded by 6.9% during 2022, largely due to regional project activity. More specifically, market conditions for the company were stable in service-led demand during 2022, led by a core segment.
3. The group described demand environment as gradually firming in replacement cycles, especially across Middle East. Notably, the group reported an energy intensity reduction of 9.9%, supported by a targeted initiative.
4. The company classified a major risk exposure as an elevated operational exposure over the period and mitigated through supplier audits. Against that backdrop, demand environment for the company were stable in service-led demand over the period, led by a core segment.
5. Demand environment for Dalmora Controls were stable in service-led demand, shaped by Middle East. Dalmora Controls reported a 7.2% efficiency gain in a key operational KPI, aided by line balancing.

System 4 retrieved chunks

1. **GOLD.** Market conditions for Solvane Logistics were stable in service-led demand, led by a core segment. Against that backdrop, a major risk exposure remained a material risk item for Solvane Logistics, prompting a mitigation program.
2. A key business driver contributed to a solidity position increase of 1.2 percentage points for the company over the period. The company described logistics bottlenecks as an elevated exposure over the period and mitigated through dual sourcing.
3. During the period, a major risk exposure remained an elevated operational risk for the company, leading to scenario planning. The company characterized the market view for the period as improving in core markets, particularly in Europe.
4. Local-for-local manufacturing served as a primary growth engine for the company through the period, especially in a core segment. By contrast, through the period, a major risk exposure remained an elevated exposure for the company, leading to selective pricing.
5. The group produced group revenue growth of 6.9%, helped by a key business driver. In addition, in its commentary, the group described the demand backdrop as improving in core markets, with emphasis on a core segment.

Example 2: N300 S_{low}

Query: What market conditions were reported for Riventor Systems in 2018 in Asia-Pacific?
Gold answer: `mixed across regions`

System 3 output: Riventor Systems reported market conditions for Asia-Pacific as mixed across regions, with the clearest activity in that region.
System 4 output: In 2018, Riventor Systems reported market conditions that were mixed across regions, with the clearest signals in OEM partners.

System 3 retrieved chunks

1. Supported by supplier decarbonization programs, Riventor Systems reported a supplier audit coverage increase of 16.8%. Market conditions for Riventor Systems were soft in project-led purchasing, most evident in Asia-Pacific.
2. **GOLD.** Riventor Systems reported an improvement in margin level of 2.2 percentage points over the period, connected to pricing discipline. The company described market conditions over the period as mixed across regions, with emphasis on Asia-Pacific.
3. Market conditions for the group were mixed across regions, with emphasis on OEM partners. The group described market conditions as mixed across regions, with the clearest activity in Asia-Pacific.
4. The group described raw material volatility as an elevated risk and countered through hedging programs. Even so, market conditions for the group were selective across customer segments, led by Asia-Pacific.
5. Management highlighted the following: Quinsera Mobility saw market conditions that were mixed across regions, particularly in Asia-Pacific. In more concrete terms, closed-loop material recovery helped Quinsera Mobility increase recycled material share by 12.7%.

System 4 retrieved chunks

1. **GOLD.** Riventor Systems reported an improvement in margin level of 2.2 percentage points over the period, connected to pricing discipline. The company described market conditions over the period as mixed across regions, with emphasis on Asia-Pacific.
2. In 2018, the company produced revenue advance of 9.3%, supported by portfolio mix improvements. In addition, the company framed the market outlook for 2018 as mixed across regions, with the clearest signals in OEM partners.
3. Through the period, the business saw market conditions that were mixed across regions, with the clearest effects in OEM partners. Through the period, its growth trajectory was helped by software-enabled offerings, most visibly in Asia-Pacific.
4. In its commentary, the group presented the market environment as moderating after a strong prior year, most evident in project-driven accounts. The group described market conditions in the period as mixed across regions, with the clearest activity in North America.
5. The group increased recycled material share by 8.3% during the period, enabled by closed-loop material recovery. Taken together, during the period, the group saw market conditions that were steady in public-sector tenders, with pressure strongest in project-driven accounts.

Figure B.12: Examples on why System 4 struggles on S_{high} despite good retrieval, compared to System 3.

C

Appendix 3 - Code and Artifacts

All code, all datasets, and all artifacts from runs in this thesis are publicly available in the below archives.

- **Zenodo archive:** `https://doi.org/10.5281/zenodo.20392650`
- **GitHub repository:** `https://github.com/junaydkader/master-thesis-code/tree/FINAL_TURN_IN`

DEPARTMENT OF MATHEMATICAL SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY