



**Università
di Genova**

DIBRIS DIPARTIMENTO
DI INFORMATICA, BIOINGEGNERIA,
ROBOTICA E INGEGNERIA DEI SISTEMI

Applications of SLMs to Custom Personal Assistants in the Healthcare Domain

Stefania Zanetta

Master Thesis

Università di Genova, DIBRIS Via Opera Pia, 13 16145 Genova, Italy
<https://www.dibris.unige.it/>



**Università
di Genova**

MSc Computer Science
Data Science and Engineering Curriculum

Applications of SLMs to Custom Personal Assistants in the Healthcare Domain

Stefania Zanetta

Advisor: G. Delzanno and G. Guerrini

Examiner: N. Noceti

March, 2025

Table of Contents

Chapter 1 Introduction	6
1.1 Problem Definition	6
1.1.1 Model Size	6
1.1.2 Answer Reliability	7
1.2 Proposed Solution	8
1.3 Thesis Outline	9
Chapter 2 Background	11
2.1 Natural Language Processing	11
2.1.1 Word Embeddings	12
2.1.2 Transformer Architecture	12
2.1.3 Large Language Models	13
2.1.4 Small Language Models	14
2.2 Retrieval-Augmented Generation	14
2.3 Edge Computing	16
Chapter 3 Architecture	17
3.1 Edge Device and Limitations	17
3.1.1 Jetson Nano	18
3.1.2 Jetson Xavier	19
3.2 Selected Models	19

3.2.1	Phi Family	20
3.2.2	Qwen Family	21
3.3	Integration of RAG	21
3.3.1	ChromaDB	22
3.3.2	Selected Word Embedding Models	22
3.3.3	ReRanker Model	23
3.3.4	HuggingFace	24
3.3.5	LangChain	24
3.4	Implementation	25
3.4.1	Knowledge Dataset	25
3.4.2	Creation of the Knowledge Base	26
3.4.3	Retriever	28
3.4.4	Prompt Template	30
3.4.5	Generation	32
Chapter 4 Evaluation		35
4.1	Evaluation Dataset	36
4.2	Information Retrieval	37
4.3	Generation	41
4.3.1	Human Evaluation	45
4.3.2	Expert Evaluation	47
4.3.3	Latency	50
Chapter 5 Other Use Cases		53
5.1	Patient History	53
5.2	Biometric Data	55
Chapter 6 Conclusions and Future Work		58

Appendix A Knowledge Base	60
Appendix B Evaluation	64
Bibliography	67

Chapter 1

Introduction

1.1 Problem Definition

Recent advances in Natural Language Processing (NLP) driven by large language models (LLMs) have led to a rapid expansion in applications, particularly in question-answering systems with impressive accuracy level. However, despite the growing adoption of LLMs and their success, in many domains they still face significant challenges, particularly in sensitive fields like healthcare. Researchers have focused on a multitude of challenges related to LLM, particularly in two major areas of concern: model size and response reliability.

1.1.1 Model Size

LLMs require substantial computational resources and hardware to run effectively, putting them beyond the reach of general users and smaller organizations, and often rely on cloud-based infrastructure, adding both expense and complexity.¹

The impact of running models of such large scale has also begun to raise concern for the environment. A study at the University of Massachusetts [SGM19] estimated the cost of training a single large model on a par with the lifetime carbon footprint of several cars. Where a single car has been estimated to produce 126,000 lbs CO₂e in a lifetime, while training one large transformer model with 213M parameters has been estimated to produce 626,155 lbs CO₂e. In this context, parameters must be understood as internal variables

¹From abstract of the paper Small Language Models: Survey, Measurements and Insights [...]compared to their large language model (LLM) counterparts, which are predominantly deployed in data centers and cloud environments.” [LLC⁺24]

that the model adjusts during training to improve its performance. Another common term is token. A token is the smallest unit of data that the model processes, such as words or characters in natural language processing.

Moreover, the need for cloud-based infrastructure to run models arises from security concerns when sensitive data are involved. In applications such as healthcare and finance, user data should be safeguarded, a priority that LLMs having to run on cloud do not always meet at the expense of the users. A model that can run locally would add a layer of protection to the user’s data, as the information remains on the device, limiting possible data exposures. Furthermore, the risks of cloud-based services do not include only breach confidentiality, which means protection of data, but also integrity, keeping the data from being modified by unauthorized users, and availability, making sure that the service is available at all time.²

In response to these issues, in recent years there has been a trend toward model compression techniques, such as quantization, and the rapid rise of Small Language Models (SLMs). SLMs can perform well on common devices, without requiring large-scale infrastructure and can be seen as a way to make machine intelligence accessible and affordable to anyone³.

1.1.2 Answer Reliability

While model compression techniques such as distillation, quantization, and pruning have enabled the development of smaller models that are more efficient and can maintain competitive performance, both Large Language Models (LLMs) and Small Language Models (SLMs) continue to struggle with answer reliability. Answer reliability refers to a model’s ability to provide accurate, current, and verifiable responses that can be related to attested sources.

One of the primary obstacles to achieving the reliability of the answer is the tendency of LLMs and SMLs to produce ‘hallucinations’. Where models generate information that are either incorrect or completely fabricated. This is due to the probabilistic nature of the models, that generates a response on learned patterns that does not always coincide with facts. Various methods have been proposed, as we can see in the survey [JLF⁺23], to mitigate hallucinations, but complete reliability remains difficult.

Another challenge is keeping the model knowledge up to date. Once language models are deployed, they cannot update their knowledge without retraining. Given also the extensive

²As reported in A Review of Machine Learning-based Security in Cloud Computing in the section “Cloud Computing Security Challenges” p. 3 [BKDN23].

³From the Introduction of Small Language Models: Survey, Measurements and Insights “A vision for SLMs is to democratize machine intelligence, making it accessible and affordable to anyone, anywhere, at any time – just like the human brain that everyone possesses.” p.1[LLC⁺24]

cost and time spent for training and the rapid evolution of the available knowledge, re-training big models is often unfeasible. Researchers started experimenting strategies such as Retrieval-Augmented Generation (RAG) [LPP⁺21, SPC⁺21], which integrates external databases into the model’s response generation process to help keep answers accurate.

RAG has been also proposed to solve the problem of traceability. As for hallucination, another problem that arises from the probabilistic nature of the models is the ability to trace the origin of the output to a reliable source. Traceability and transparency are of the utmost importance, especially in sensitive fields such as law and healthcare. In these cases, an unverified or misleading response could lead to significant consequences. Retrieval Augmented Generation can trace the knowledge from which the statement has been generated, but the integration of retrievals from external knowledge, makes the models resource-intensive in large-scale applications adding a problem of scalability [GXG⁺24].

1.2 Proposed Solution

Given the outlined challenges in using language models in healthcare, this thesis proposes a solution that leverages the capabilities of Small Language Models (SLMs), deployable on edge devices, in combination with the strengths of a Retrieval-Augmented Generation (RAG) system that sources information from a controlled, reliable dataset. The goal is to develop a chatbot that provides guidance on first aid in emergencies where medical professionals are unavailable and traditional communication devices may be inaccessible. In such scenarios, having access to immediate and accurate first-aid instructions can significantly improve a victim’s chances of survival while awaiting medical help.

An immediate response or being able to follow simple safety measures can yield significant positive outcomes. For instance, in cases of cardiac arrest, the European Resuscitation Council emphasizes that “the immediate initiation of CPR can double or quadruple survival after cardiac arrest”⁴, underlining the critical impact of bystander intervention in emergencies. Similarly, the “Seizure related injuries - Frequent injury patterns, hospitalization and therapeutic aspects” states that “patients with epilepsy are more frequently admitted to hospitals following an injury”⁵, emphasizing that even simple actions such as removing nearby objects to prevent injury, can significantly reduce the risk of complications.

It is important to note that first aid procedures are largely standardized across many countries, with key guidelines provided by international organizations such as the International Liaison Committee on Resuscitation (ILCOR) [ILC].

⁴From the section Cardiac arrest subsection 2: Early bystander CPR p.6 [MNB⁺15]

⁵From the Introduction p.2 [MSM⁺22]

However, despite this standardization, it is also crucial to emphasize that legislation regarding first aid can differ from country to country. For example, while the procedures for CPR or the treatment of burns may be equal, the legal frameworks, such as Good Samaritan laws, vary between nations [Wik24].

Therefore, this thesis will reference and utilize manuals and guidelines from various countries, such as United States and Canada, in order to present a comprehensive understanding of first aid practices.

1.3 Thesis Outline

The following chapters of this thesis will explore the theoretical foundations, technical details, implementation aspects, and evaluation of the proposed system. The work examines both the challenges and opportunities of deploying small language models (SLMs) in resource-constrained environments while maintaining reliable and contextually accurate responses. The structure of the thesis is as follows:

- **Chapter 2:** This chapter introduces the key concepts basis of the thesis. It provides an overview of fundamental NLP topics, including word embeddings, the Transformer architecture, and Large Language Models (LLMs), with a focus on their limitations. The chapter then presents Small Language Models (SLMs) as a resource-efficient alternative, followed by a discussion on Retrieval-Augmented Generation (RAG), specifically the Naive RAG approach used in this work. Finally, it examines the role of edge computing in deploying AI models in resource-constrained environments.
- **Chapter 3:** Building on the concepts introduced in Chapter 2, this chapter details the architecture of the RAG system. It first explores edge computing devices and their constraints before discussing the small language models selected. The chapter then introduces the key tools used for RAG implementation, followed by a detailed description of how the system was constructed.
- **Chapter 4:** This chapter focuses on assessing the performance of the proposed RAG system. It begins by outlining the evaluation dataset and then examines the effectiveness of the retrieval phase, including an analysis of document chunking strategies and embedding model performance. The generation phase is then evaluated using both automated metrics and human assessments, including expert reviews. Finally, the computational efficiency of the system is analyzed, particularly in terms of response latency.
- **Chapter 5:** This chapter explores additional potential applications of RAG in the

healthcare domain. While these use cases are conceptual rather than fully implemented, they demonstrate the versatility of the proposed approach.

- **Chapter 6:** The thesis concludes with a summary of findings, highlighting the contributions of the study and discussing potential directions for future research and improvements.

Chapter 2

Background

This chapter provides an overview of the key concepts for the development of Small Language Models (SLMs) with Retrieval-Augmented Generation (RAG).

First, it introduces foundational topics in NLP, such as word embeddings, the Transformer architecture, and Large Language Models (LLMs), providing information for understanding how modern language models process and generate text. Particularly, the discussion of LLMs will underline their limitations, such as their high computational demands and environmental impact. These challenges highlight the need for alternatives, leading to the introduction of Small Language Models (SLMs), which aim to provide effective language understanding and generation while using significantly fewer resources.

Section 2.2 then introduces Retrieval-Augmented Generation (RAG), a technique that enhances language models by integrating external knowledge sources to improve accuracy, focusing on the Naive RAG methodology that will be implemented in this thesis. In the end, in Section 2.3, we explore edge computing in deploying AI models in resource-constrained environments.

2.1 Natural Language Processing

Natural Language Processing (NLP) is a subfield of artificial intelligence that focuses on enabling machines to understand, interpret, and generate human language. Over the years, NLP has evolved significantly, from rule-based approaches to statistical methods and, more recently, deep learning models.

Traditional NLP techniques relied on hand-crafted rules and lexicons, which, while effective in narrow domains, struggled with scalability and ambiguity. The advent of deep

learning revolutionized NLP by automating feature extraction and enabling models to capture complex linguistic patterns. Techniques such as word embeddings, and more recently, transformer-based architectures, have significantly enhanced NLP applications, paving the way for modern language models.

2.1.1 Word Embeddings

In natural language processing, the task of representing words as vectors has a long history [QSX⁺21]. Traditionally, natural language models used sparse representations, such as one-hot encoding, where each word is represented by a vector as long as the vocabulary size, with a single non-zero entry corresponding to the word's index. Word embeddings [MCCD13] represent a key innovation in the field offering a dense, low-dimensional representation of words in continuous vector space. These vector representations capture semantic relationships, allowing semantically similar words to have similar vector representations, properties that one-hot encoding fails to capture.

Word embeddings are learned from large text corpora and have become a foundational tool in various NLP tasks such as sentiment analysis [MCCD13], machine translation [CLR⁺18] and laying the groundwork for contextual embeddings [PNI⁺18] and transformer architecture [VSP⁺23].

2.1.2 Transformer Architecture

The introduction of the Transformer architecture by Vaswani [VSP⁺23] marked a shift in how NLP models were designed and trained. As illustrated in Fig. 2.1, the Transformer architecture consists of two main components: an encoder and a decoder.

Unlike recurrent neural networks (RNNs) and long short-term memory (LSTM) networks, which process text sequentially, the Transformer model processes all words in a sequence simultaneously through a mechanism called self-attention. This parallelization significantly improves training time, especially on large datasets, and enhances the model's ability to handle longer sequences compared to previous sequential models. This mechanism allows the model to learn contextual relationships between words and the information about the order of words in a sequence, leading to a substantial improvement in performance across a wide range of tasks.

The Transformer architecture has proven to be a core innovation in NLP, serving as the basis for many state-of-the-art models, such as BERT and GPT.

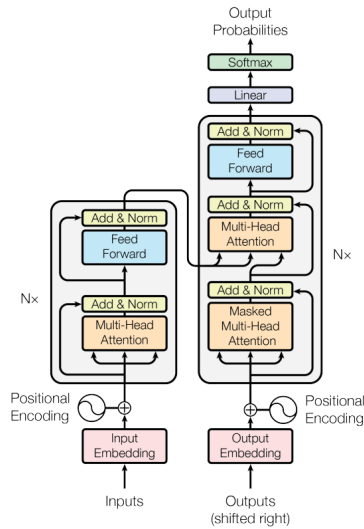


Figure 2.1: Transformer Architecture. (Vaswani, 2023) [VSP⁺23]

2.1.3 Large Language Models

Large Language Models (LLMs) represent a significant leap forward in the capacity and capabilities of NLP systems.

As the name suggests, the scale of LLMs is one of their most distinguishing characteristics. GPT-3, for example, boasts 175 billion parameters, an order of magnitude greater than its predecessor GPT-2, which had 1.5 billion parameters [Ope]. Trained on large datasets, which include diverse sources like books, websites, and other textual data, the massive number of parameters allows LLMs to capture more complex patterns in language, representing an expansive range of linguistic, contextual, and factual information.

One key aspect of LLMs is their ability to generalize from the vast amount of data they have been trained on, enabling them to perform a range of NLP tasks, such as question answering, conversational agents and content generation.

Despite these promising applications, LLMs are not without their challenges and concerns. One major issue is their potential to propagate bias present in the data on which they are trained. Since these models are exposed to a vast range of text, they can inadvertently learn and reproduce harmful stereotypes, misinformation, or biased behavior. Efforts to mitigate these biases have become a focal point of research, but ensuring fairness and remains a difficult challenge [BGMMS21].

Another concern is the environmental cost associated with training such large models. The computational resources required to train large models can lead to substantial carbon

footprints, raising ethical questions about the environmental impact [SGM19].

2.1.4 Small Language Models

In response to the increasing computational cost and ethical concerns associated with LLMs, the rise of Small Language Models (SLMs) represents an emerging trend aimed at achieving similar performance with significantly reduced resource consumption. Since the end of 2023 [LLC⁺24], the number of SLM models has surged significantly, as depicted in Fig. 2.2; although there is no universally accepted definition of Small Language Models (SLMs), we refer to the definition provided in the survey [LLC⁺24]: SLMs are decoder-only transformer architectures with a weight range of 100M to 5B parameters.

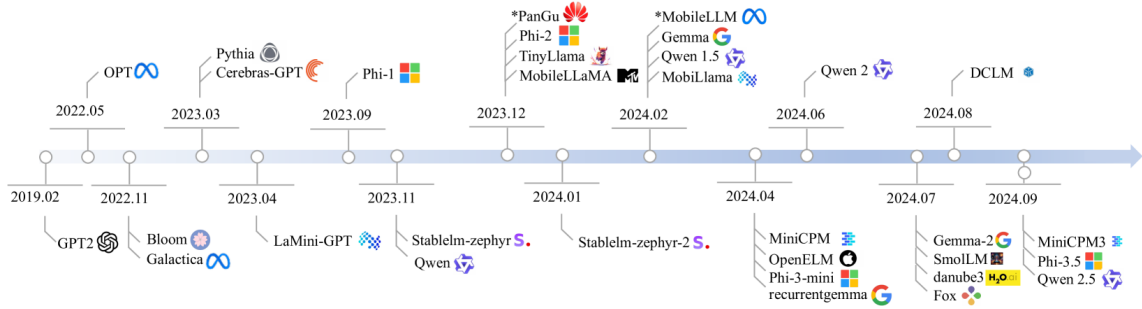


Figure 2.2: An overview of SLMs. (Zhenyan 2024) [LLC⁺24]

Techniques like knowledge distillation, pruning, and parameter sharing, allow SLMs to retain much of the performance of larger models while using fewer parameters [AVS24] [SLBK24] [WCL⁺24]. By reducing model size and improving efficiency, SLMs enable the deployment of advanced NLP models where computational resources are limited, such as mobile devices or edge computing environments. Moreover, smaller models may alleviate the environmental concerns associated with training large models, offering a more sustainable pathway for future NLP developments.

2.2 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) was introduced in 2021 [LPP⁺21] as a method to enhance task-specific Natural Language Generation (NLG) by integrating external, non-parameterized knowledge into Large Language Models (LLMs). This approach offers a solution for addressing the limitations of LLMs, particularly in terms of their reliance on parameterized knowledge.

Parameterized knowledge refers to the implicit knowledge embedded in a model during its training phase. This knowledge is encoded within the model’s parameters, allowing it to generate responses based on patterns learned from the training data. While this type of knowledge is accessible without external memory, it can be limited, and models may fail to provide accurate or relevant insights, especially when they generate “hallucinations” outputs that are factually incorrect, misleading, or nonsensical.

In contrast, non-parameterized knowledge refers to external information sources, often stored in databases, that are not inherently part of the model. Unlike parameterized knowledge, non-parameterized knowledge can be updated, expanded, and inspected in real-time. This makes it an ideal supplement to LLMs, allowing models to access external, dynamically adding information, and improving their ability to produce more accurate, context-aware, and up-to-date responses.

Over time, researchers have developed and refined various RAG-based approaches [GXG⁺24]. This thesis will focus on the “Naive RAG” methodology, which offers a straightforward yet effective implementation of the RAG concept.

As outlined in the [GXG⁺24], the process of a Naive RAG system can be broken down into three core steps: indexing, retrieval, and generation. These steps work in tandem to leverage external knowledge during the generation of a response.

- **Indexing:** The first step in a RAG system is creating a knowledge database. Initially, data from various formats is converted into plain text. This text is then divided into smaller, manageable segments, known as chunks. Each chunk is encoded into a vector representation using an embedding model, and the vectors are stored in a vector database for efficient retrieval.
- **Retrieval:** When a query is received, it is encoded into a vector using the same embedding model that was used during the indexing phase. The query vector is then compared to the vector representations of the pre-existing chunks in the database, and the system computes a similarity score for each chunk. The chunks with the highest similarity scores are selected as the most relevant sources of external knowledge, expanding the model’s contextual understanding and guiding the generation of a response.
- **Generation:** Using the selected context from the retrieved chunks, the model generates a response to the query. This process combines the model’s internal knowledge with the external, contextually relevant information to provide a more accurate and informative answer.

2.3 Edge Computing

Interest in AI continues to grow, along with the demand for reduced latency and improved real-time data processing. Unlike cloud computing, which processes data in centralized data centers, edge computing processes data closer to its source, at the edge of the network. This approach not only reduces latency but also enhances security and privacy, as data remains localized, on devices or sensors, rather than being transmitted to distant data centers. Traditional cloud computing struggles to handle massively distributed applications and analyze their data effectively due to the limitations in network capacity when tasks are moved to the cloud [BJCG21]. For example, applications such as autonomous driving demand strict latency requirements, which the cloud often cannot meet due to its physical distance from end users.

Edge devices vary in form, from large servers to low-power System-on-a-Chip devices like the Raspberry Pi and Jetson Nano [BJCG21]. Although these devices have limited computing power and hardware resources, they are often equipped with accelerators that boost performance, optimizing them for edge computing environments.

Additionally, Edge AI enables applications to operate efficiently even in areas with unreliable or intermittent connectivity by processing data directly at the source [GGH⁺24]. This capability is crucial in remote settings where consistent and stable internet access cannot always be ensured, making it particularly critical in domains such as healthcare.

Chapter 3

Architecture

In the previous chapter, the components required for the development and deployment of a Retrieval-Augmented Generation system on an edge device were discussed. This study greatly benefited from the collaboration of BSc Nicolò Guainazzo, with whom the initial tests on the Jetson Nano were conducted in parallel. Additionally, the selection of the Small Language Model and part of the structuring of the RAG system were discussed and refined together.

This chapter proceeds with a more detailed exploration, beginning with Section 3.1, which introduces the available edge computing devices, emphasizing their inherent limitations.

Section 3.2 focuses on the selection of the small language model chosen for the thesis, outlining for each its key characteristics.

In Section 3.3, an in-depth overview of the primary tools employed for the RAG system is provided, highlighting their functionalities and relevance to the overall implementation.

Finally, in Section 3.4, the implementation of the RAG system is presented, incorporating the components and tools discussed.

3.1 Edge Device and Limitations

As previously mentioned, edge computing comes with several limitations tied to the specific devices used. These constraints play a significant role in shaping key decisions during the implementation of the RAG system, often leading to various trade-offs.

One of the most prominent families of edge devices for AI is the NVIDIA Jetson series. The introduction of the Jetson Nano in 2019 marked a significant shift, catering to both devel-

opers and enthusiasts by offering a machine tailored for AI at an affordable price [Uch19]. While earlier Jetson products were primarily used for vision computing and robotics, the Nano expanded its scope by including libraries for deep learning, computer vision, and multimedia processing. As expected from NVIDIA, Jetson devices are powered by robust GPUs and CUDA-X accelerated libraries. With its relatively low power consumption, the Jetson Nano represented an important step forward in bringing GPU-based hardware to the edge, given the typically power-hungry nature of GPUs.

This thesis examines two NVIDIA edge devices: the Jetson Nano 4GB and the Jetson Xavier AGX. The next sections detail the unique limitations of both devices, some of which could not be overcome, as well as the challenges that will shape the implementation of the RAG system.

3.1.1 Jetson Nano

As stated before, NVIDIA announced the Jetson Nano 4GB in March 2019, introducing it as an affordable and accessible edge AI computing platform. Upon its release, it was well-received due to its low cost and the ability to run deep learning models at the edge, see, e.g., [Ass19] [PP19] [Mak19] [Alt19]. It features a quad-core ARM Cortex-A57 processor operating at 1.43 GHz, paired with a 128-core Maxwell GPU and it's equipped with 4GB of LPDDR4 RAM [NVIb]. However, despite of its promising specifications, experiments demonstrated significant limitations for modern AI models.

One of the major problems was software compatibility. NVIDIA provides Software Development Kits (SDKs) called JetPacks for its Jetson devices. In 2021 it was announced that the Jetson Nano will no longer receive updates for the JetPack SDK, concluding its compatibility with version 4.6.5 [suh21]. This version limits the device to CUDA 10.2 and Python 3.6, making it impossible to install newer versions of PyTorch, since version 2.0+ requires CUDA 11 and version 1.11+ requires Python 3.7. This constraint severely limits the device's compatibility with other key AI libraries, such as Hugging Face's Transformers.

Further tests confirmed additional limitations: Ollama, a popular framework for running language models, defaults to CPU execution on the Nano, as its CUDA is too outdated [Res]. Attempts to use NVIDIA's 'jetson-containers' repository also failed, as it requires JetPack 5 or higher.

Ultimately, as the company has shifted focus to other devices leaving the Jetson Nano increasingly obsolete, the lack of support for CUDA 11, newer PyTorch versions, and key AI frameworks has rendered it impractical for modern language models.

As mentioned in the previous section, the tests on the Jetson Nano were conducted in collaboration of BSc Nicolò Guainazzo.

3.1.2 Jetson Xavier

Released a year before of the Jetson Nano, on September 12, 2018, NVIDIA announced the release of the Jetson AGX Xavier, a high-performance edge AI computing platform, designed primarily for robotics and autonomous machines [Nef18].

The Jetson AGX Xavier is powered by an 8-core NVIDIA Carmel 64-bit ARMv8.2 CPU and a 512-core NVIDIA Volta GPU, equipped with 64 Tensor Cores. It includes 16GB of 256-bit LPDDR4x memory and 32GB of eMMC 5.1 storage [NVI21]. As the specifications are higher than Jetson Nano, so its price, levitating from 99\$ to an expensive 2,499\$ making it significantly less affordable than its small counterpart. One of the standout architectural features of the Jetson AGX Xavier is its unified memory architecture, which allows the CPU and GPU to share the same memory. Unlike conventional systems where the GPU requires separate memory and must rely on memory transfers, this design improves on these bottlenecks as well as reduced power consumption [MSV19].

As with other members of the Jetson family, the AGX Xavier is supported by JetPack SDK. However, unlike the Jetson Nano, which is limited to JetPack 4.6.5, the AGX Xavier benefits from more extended support, concluding its compatibility with JetPack 5.1.4 [NVIa]. This version incorporates more recent libraries, such as CUDA 11.4. Extending the compatibility to Pytorch 2.0 and Python 3.8. This update makes a substantial difference: unlike the Jetson Nano, it allows the use of key libraries, such as Hugging Face and LangChain, and enables full utilization of the GPU.

Since the GPU on the Jetson Nano could not be fully utilized and lacked support for critical libraries, the RAG system was tailored for the Jetson AGX Xavier, leading to the discontinuation of the Jetson Nano in this project. From this point forward, most design decisions in the system’s implementation will be dictated by the constraints of the Jetson AGX Xavier.

3.2 Selected Models

Selecting the Small Language Models (SLMs) posed a challenge due to the wide range of available options. As previously mentioned, the selection process followed the definition of SLMs provided by the survey [LLC⁺24], considering only 100M to 5B parameters, with a focus on models that offer high performance while maintaining a compact size.

To refine the selection, three prominent SLM families were considered: Phi (Microsoft), Qwen (Alibaba), and Gemma (Google). Each model family was chosen for specific reasons: Gemma was designed with a strong emphasis on safety, incorporating reinforcement learning from human feedback [BW24]; Phi for its performance, surpassing even larger models

on reasoning tasks [Bil24]; and Qwen provided flexibility in model size, with versions as small as 0.5 billion parameters, making it well-suited for resource-constrained environments [Tea24d].

Additionally, only instruction-tuned models were considered. Instruct models are language models that have been fine-tuned using supervised learning and reinforcement learning techniques to better follow human instructions. These models are optimized for tasks that require step-by-step reasoning and question answering.

However, due to the limitations of the Jetson AGX Xavier, Gemma had to be excluded, as it requires a newer version of PyTorch that is not supported by JetPack 5.1.2.

3.2.1 Phi Family

Microsoft introduced the Phi family of Small Language Models (SLMs) starting with Phi-1 in June 2023 [GZA⁺23]. The Phi-1 model was designed primarily for Python code generation. Later that year, Microsoft expanded its work with Phi-1.5, which introduced improvements for question answering. In addition to the original data sources used for Phi-1, Phi-1.5 incorporated various synthetic NLP datasets, enhancing its ability to handle more diverse language tasks [LBE⁺23]. Building on this progress, Microsoft released Phi-2, a 2.7 billion-parameter model, which improved reasoning and language understanding while nearly doubling the 1.4 billion parameters of Phi-1.5.

In April 2024, the Phi-3 family was introduced, consisting of three models: Phi-3-mini (3.8B parameters), Phi-3-small (7B parameters) and Phi-3-medium (14B parameters). These models underwent rigorous safety evaluations, including sensitive use review, and security assessments, ensuring responsible development and deployment. The Phi-3 models significantly outperformed language models of similar and even larger sizes on key benchmarks, such as MMLU (Massive Multitask Language Understanding) and CommonsenseQA [Bil24]. MMLU evaluates a model’s performance across a wide range of subjects, while CommonsenseQA tests a model’s ability to answer questions requiring general reasoning and common sense knowledge.

Later in 2024, Microsoft expanded the Phi-3 family with the introduction of Phi-3.5, which included: Phi-3.5-mini, a multimodal model Phi-3.5-vision and Phi-3.5-MoE (Mixture of Experts). Phi-3.5-mini introduced key improvements, including multilingual support and, most importantly, long-context performance. This model excelled in long-document summarization, question answering based on long texts, and information retrieval, making it well-suited for applications requiring extended context understanding [Adi24].

For this project, the selected models from the Phi family were: Phi-3-mini-instruct and Phi-3.5-mini-128k-instruct.

3.2.2 Qwen Family

Introduced in September 2023, Qwen debuted with four specialized models: Qwen, designed for general conversation and fine-tuned using techniques like Reinforcement Learning from Human Feedback (RLHF); Code-Qwen, trained on a large dataset of code; Math-Qwen, focused on mathematical problem-solving; and Qwen-VL, a multimodal model capable of processing both text and images [BBC⁺23]. Initially, Qwen was available in three sizes: 1.8B, 7B, and 14B parameters. Later, in 2024, a larger 72B variant was introduced. Since its first iteration, Qwen was multilingual, with a primary focus on Chinese and English, thanks to the diverse dataset it was trained on.

In February 2024, Qwen1.5 was announced, introducing a new series of models with eight different sizes, ranging from 0.5B to 72B parameters. This version featured several key improvements, including an extended context length, to 32,000 tokens, and enhanced multilingual capabilities, expanding its performance across 12 languages [Tea24b].

Later that year, Qwen2 was released, further improving upon its predecessor by increasing the supported context length and expanding multilingual support to 27 languages. More notably, Qwen2 introduced instruction-tuned variants in five different model sizes [Tea24a].

Shortly afterward, the Qwen2.5 series was launched, significantly scaling up the training process to incorporate 18 trillion tokens. This expansion led to improved factual knowledge, enhanced reasoning capabilities, and superior performance across multiple benchmarks, including general knowledge (MMLU), coding (HumanEval), and mathematical reasoning (MATH). Additionally, Qwen2.5 demonstrated improved comprehension and manipulation of structured data formats such as tables and JSON, making it more effective for applications requiring precise data extraction, transformation, and interpretation [Tea24c].

For this project, the selected models from the Qwen2.5-Instruct family were Qwen2.5-0.5B-Instruct, Qwen2.5-1.5B-Instruct, and Qwen2.5-3B-Instruct.

3.3 Integration of RAG

As previously outlined, a Retrieval-Augmented Generation system consists of three primary stages: Indexing, Retrieval, and Generation. This section provides an overview of the key components involved in each of these stages before delving into the specific implementation details.

The initial step of a RAG system involves the construction of a knowledge database. Section 3.3.1 introduces ChromaDB, the selected vector database library, which facilitates the storage and retrieval of text embeddings. Section 3.3.2 discusses the two Word Embedding models tested for encoding text chunks to be stored in the database.

For the retrieval phase, Section 3.3.3 presents a ReRanker model, which was employed to refine the retrieval process, improving the context that will be processed for response generation.

In the final step, Generation, the system leverages the small language models previously introduced to produce context-aware responses. Two critical components are utilized to support this stage: Section 3.3.4 details the Hugging Face platform, which provides access to the language and word embedding models, while Section 3.3.5 covers LangChain, which integrates the elements of the RAG pipeline.

3.3.1 ChromaDB

A key component of a Retrieval-Augmented Generation (RAG) system is the vector database, which enables the efficient retrieval of relevant documents based on semantic similarity rather than exact keyword matching.

Vector databases store embeddings, numerical representations of text, that facilitate similarity searches using distance metrics such as cosine similarity or Euclidean distance. These embeddings allow RAG systems to retrieve contextually relevant information prior to generating responses, thus enhancing both the accuracy and coherence of the outputs.

One of the most widely used libraries for vector search is Facebook AI Similarity Search, FAISS, developed by Meta AI for the efficient storage and querying of dense vectors [Sim24]. However, due to FAISS not supporting ARM architecture builds [fac22], an alternative library compatible with the Jetson Xavier AGX platform is required. In this context, we have selected ChromaDB library.

ChromaDB is an open-source vector database optimized for machine learning applications, particularly those involving RAG systems [Nid24]. It provides persistent storage for embeddings and integrates with LangChain and Hugging Face pipelines, thus enabling efficient document retrieval for generative AI models [Lan23]. For our implementation on the Jetson Xavier AGX, we utilize version chromadb 0.3.29.

3.3.2 Selected Word Embedding Models

As previously discussed, word embedding models convert text into numerical representations that capture the semantic relationships between words and form the foundation of the database using ChromaDB. Given the medical nature of the data, two word embedding models were tested: bge-small-en-v1.5 and MedEmbed-small-v0.1.

- **bge-small-en-v1.5 model**¹, developed by the Beijing Academy of Artificial Intelligence, BAAI, is a compact yet effective embedding model designed for semantic search and dense retrieval. It produces high-quality vector representations optimized for efficient document retrieval, making it well-suited for Retrieval-Augmented Generation based applications [Sis23]. This model strikes a balance between performance and computational efficiency, allowing it to function effectively on resource-constrained devices, such as the Jetson Xavier AGX.
- **MedEmbed-small-v0.1 model**², developed as part of the MedEmbed family, is a fine-tuned version of the BAAI/bge-small-en-v1.5 model specifically optimized for medical and clinical data. Unlike general-purpose models, MedEmbed utilizes a vast collection of clinical notes and medical literature from PubMed to enhance performance on medical retrieval tasks. MedEmbed-Small has demonstrated superior performance over larger general-purpose models like BAAI/bge-small-en-v1.5 on various medical retrieval benchmarks, including MedicalQARetrieval, PublicHealthQA, and TRECCOVID [Bal24].

3.3.3 ReRanker Model

In Retrieval-Augmented Generation systems, the retrieval of the most relevant documents is essential for ensuring the generation of high-quality responses by the language model. A reranking model enhances the search process by re-evaluating the relevance of the retrieved documents with respect to the user’s query. Unlike traditional retrieval methods that rely primarily on surface-level similarity metrics, rerankers leverage deeper semantic understanding to reorder documents based on their contextual relevance [Mau24].

In the context of an RAG pipeline, the reranker functions act as a second-pass filter, refining the initial set of retrieved documents to prioritize those that are most contextually relevant to the query. In contrast to faster similarity search techniques, rerankers are computationally more demanding [Mau24], as they generate scores through direct pairwise comparisons of query-document pairs, making them significantly slower than vector-based similarity methods. As a result, it is important to limit the number of comparisons performed. Therefore, the reranker operates after the initial retrieval step, which gathers a broad set of candidate documents, to perform a finer-grained analysis of their relevance based on a deeper understanding of the query’s intent.

Various types of reranking models exist, including multi-vector models [Bha24]. For this thesis, the selected model is the cross-encoder model bge-reranker-v2-m3³. A cross-encoder

¹<http://huggingface.co/BAAI/bge-small-en-v1.5>

²<https://huggingface.co/abhinand/MedEmbed-small-v0.1>

³<https://huggingface.co/BAAI/bge-reranker-v2-m3>

is a neural network architecture that processes a pair of text inputs, in this case query and document, and generates a single similarity score, indicating their relevance to one another [Mud23].

3.3.4 HuggingFace

Hugging Face is an open-source platform for data science and machine learning, widely recognized as a central hub for AI development.

The Hugging Face Hub hosts over 900,000 models and 200,000 datasets [hug22], all of which are open-source and publicly accessible. This platform enables users to browse, test, and integrate pre-trained models into their applications. Hugging Face offers a diverse range of AI models, each one denoted by a task-specific pipeline [hug18], including:

- **Natural Language Processing:** Tasks such as translation, summarization, text generation, sentiment analysis, and text embedding.
- **Audio Processing:** Capabilities including automatic speech recognition, voice activity detection, and text-to-speech conversion.
- **Computer Vision:** Tasks like image classification, depth estimation, and image-to-image transformations.
- **Multimodal Models:** Advanced AI systems that integrate multiple data types, text, images, and audio, to generate outputs.

The Transformers library facilitates interaction with models, allowing users to send tasks and receive results. Additionally, models can be downloaded and fine-tuned using custom datasets. Due to its extensive library and user-friendly interface, Hugging Face has become a leading platform for the deployment and fine-tuning of Large Language Models and Small Language Models.

3.3.5 LangChain

LangChain is an advanced open-source framework designed to streamline the development, deployment, and optimization of Large Language Model based applications. It provides a suite of tools that enable developers to efficiently build applications by leveraging pre-built components and third-party integrations [Lan24]. LangChain is particularly suited for the development of Retrieval-Augmented Generation and agent-based applications.

The LangChain framework comprises several core open-source libraries, including:

- **langchain-core:** Provides foundational components for chat models [Lan24].
- **langchain:** Offers higher-level constructs that facilitate the creation of chains, agents, and retrieval strategies for AI applications [Lan24].
- **langchain-community:** A repository of third-party integrations [Lan24].

A key example of its third-party integrations is Hugging Face, which allows users to utilize an extensive library of embedding models critical for tasks such as document retrieval, similarity search, and knowledge-based AI applications[hug25].

Additionally, as previously mentioned, LangChain supports integration with the ChromaDB library[Lan23], facilitating the connection between Hugging Face’s embedding models and Chroma’s vector database. This integration enables developers to leverage Hugging Face’s word embeddings to construct local vector databases and facilitates the incorporation of Retrieval-Augmented Generation systems.

3.4 Implementation

In this section, the detailed implementation of the Retrieval-Augmented Generation system is presented, outlining the key components and their integration.

Section 3.4.1 provides an overview of the document sources that form the basis of the vector database. Section 3.4.2 describes the construction of the vector database using ChromaDB, explaining the data processing steps. Section 3.4.3 focuses on the implementation of the retrieval mechanism, detailing the integration of both the retriever and reranker models to optimize information retrieval. Section 3.4.4 explores the design of system prompts, explaining how queries are structured and how the retrieved context is incorporated for each model. Finally, Section 3.4.5 presents the full implementation of the RAG pipeline, integrating the retrieval mechanism and the language model.

3.4.1 Knowledge Dataset

A fundamental component of the RAG system is the knowledge base, which serves as the primary source of information for retrieval during the query process. In this implementation, the knowledge base is composed of a curated set of open-source documents, specifically manuals and study materials of first aid. These documents provide authoritative guidance on medical emergencies, ensuring the system retrieves accurate and reliable information.

The dataset consists of the following documents:

- **Manual of St. John Ambulance Canada:** This manual is a widely recognized resource for first aid training across Canada. It provides detailed guidelines on responding to injuries and medical emergencies, including step-by-step instructions, wound care, and managing other health crises. Designed to be accessible, the manual presents information in a structured, easy to navigate and user-friendly format. An example of manual page is provided in appendix A.1. ⁴
- **Study Materials from the National CPR Association:** The National CPR Association provides a comprehensive collection of study guides, manuals, and tests to practice exams covering a broad range of emergency response topics. Their resources align with the latest 2024 American Heart Association (AHA) and Red Cross guidelines and are designed to prepare individuals for various certifications. An example of manual page is provided in appendix A.2. ⁵
- **First Aid Reference Notes:** By the company Actual First Aid Training, they provide various courses for first aid. These notes provide a concise yet thorough summary of essential first aid procedures, covering a broad spectrum of medical emergencies such as choking, burns, fractures. Designed for quick reference, they offer practical guidance in urgent situations, ensuring that first aiders can retrieve critical information efficiently. An example of manual page is provided in appendix A.3. ⁶

3.4.2 Creation of the Knowledge Base

Before implementing the Retrieval-Augmented Generation system, the first essential step involves building the vector database.

The `DirectoryLoader` is utilized to scan the directory `DIRECTORY_PDF` where the previously presented documents are stored. It extracts text from all PDF files within it. The `glob="/*.*pdf"` argument ensures that only PDF files are selected, while `PyPDFLoader` is specified as the document parser, responsible for extracting raw text content from the PDF files.

Once the documents have been loaded into memory, they must be segmented into smaller chunks. This is accomplished using the `RecursiveCharacterTextSplitter` class, which partitions the text into segments of a predefined `N_CHUNKS` size. The chunk sizes selected for testing are 500, 1000 and 1500 tokens. In the context of focusing on first aid procedures, the chunk size must be chosen to be sufficiently large to avoid cutting off critical

⁴Available at: https://sja.ca/en/digital_first_aid_reference_guide_preview

⁵Available at: <https://www.nationalcprassociation.com/free-cpr-study-guide>

⁶Available at: <https://actualfirstaid.com/first-aid-notes--feedback.html>

information. Furthermore, a small overlap `N_OVERLAP` of characters between consecutive chunks is introduced to ensure continuity of information across adjacent segments. For this implementation, an overlap of 10% of the chunk size is selected, corresponding to 50, 100 and 150 tokens, respectively.

```
EMBD_MODEL="BAAI/bge-small-en-v1.5"
DIRECTORY_DB='BGE_DB'
DIRECTORY_PDF="/PDF"
N_CHUNKS=500
N_OVERLAP=50

def wrap_text_preserve_newlines(text, width=110):
    """
    Wraps each line of the input text while preserving
    newline characters.
    This function splits the input text into lines
    at newline characters,
    wraps each line to ensure it doesn't exceed the specified width.

    Parameters:
    - text (str): The input text to be wrapped.
    - width (int, optional): The maximum width for each line.
    Default is 110.

    Returns:
    - str: The wrapped text with lines not exceeding the specified
    width, while preserving the original newline characters.
    """
    # Split the input text into lines based on newline characters
    lines = text.split('\n')

    # Wrap each line individually
    wrapped_lines = [textwrap.fill(line, width=width) for line in lines]

    # Join the wrapped lines back together using newline characters
    wrapped_text = '\n'.join(wrapped_lines)

    return wrapped_text

# Load and process the text files
```

```

loader = DirectoryLoader(DIRECTORY_PDF, glob="./*.pdf",
    loader_cls=PyPDFLoader)
documents = loader.load()

# Break the loaded documents into smaller chunks
text_splitter = RecursiveCharacterTextSplitter(chunk_size=N_CHUNKS,
    chunk_overlap=N_OVERLAP)
texts = text_splitter.split_documents(documents)

# Initialize the HuggingFaceBgeEmbeddings model
instructor_embeddings = HuggingFaceBgeEmbeddings(model_name=EMBD_MODEL,
    model_kwargs={"device": "cuda"})

# Supplying a persist_directory will store the embeddings on disk
vectordb = Chroma.from_documents(documents=texts,
    embedding=instructor_embeddings,
    persist_directory=DIRECTORY_DB)

```

The divided text is converted into numerical embeddings, the `HuggingFaceBgeEmbeddings` class is used for this task, with the specific embedding model selected based on the `EMBD_MODEL` parameter. As previously introduced, the embedding models selected `bge-small-en-v1.5` and `MedEmbed-small-v0.1`.

Once the embeddings are generated, they are organized and stored in a structured format to facilitate efficient querying. The `Chroma.from_documents` method is utilized to create a vector database from the processed text chunks. The vector database is then saved to disk in the directory `DIRECTORY_DB`.

3.4.3 Retriever

Once the vector database has been created and stored, the Retrieval-Augmented Generation system can be initiated, beginning with the retrieval component.

The first step is loading the vector database. The `Chroma` class instantiates a vector database from the directory, `PERSIST_DIRECTORY_DB`, where embeddings have been stored.

The `embedding_function` utilizes `HuggingFaceBgeEmbeddings` to generate embeddings for the retrieval of documents. The model name, specified by `DB_EMBED_MODEL_NAME`, refers to the embedding model used, the same used to embed the vector database. While `model_kwargs` contains additional configuration. `DEVICE_MAP` is set to `cuda` to acceler-

ate processing.

After the database loading, a retriever is created from the vector database. The method `as_retriever` transforms the vector database into a retriever that can be queried for relevant documents based on a given input. The `search_type` defines the specific search strategy to be used. The selected method for testing was `similarity_score_threshold`. Meanwhile `k` specifies the number of documents to retrieve. The value of `RETRIEVED_K` was set to 5.

```
# Initialize variables from 'config.py'
PERSIST_DIRECTORY_DB='BGE_DB'
DB_EMBED_MODEL_NAME="BAAI/bge-small-en-v1.5"
RERANKER_MODEL="BAAI/bge-reranker-v2-m3"
DEVICE_MAP='cuda'
RETRIVED_K=5
SEARCH_TYPE_RETRIEVER="similarity_score_threshold"
SIMILARITY_THRESHOLD=0.6
RETRIEVER_TOP_N=2

# Initialize the prompt based on model passed in arg
prompt_template = process_prompt(args)
# The function will return the model ID
model_id = initialize(args)

# Load the existing vector database from the specified directory
# (PERSIST_DIRECTORY_DB)
# using the embedding function (HuggingFaceBgeEmbeddings)
# used during storage.
vectordb = Chroma(
    persist_directory=PERSIST_DIRECTORY_DB,
    embedding_function=HuggingFaceBgeEmbeddings(
        model_name=DB_EMBED_MODEL_NAME,
        model_kwargs={"device": DEVICE_MAP})
)

# Create a retriever from the loaded vector database
retriever = vectordb.as_retriever(
    search_type=SEARCH_TYPE_RETRIEVER,
    search_kwargs={
        "k": RETRIEVED_K,
        "score_threshold": SIMILARITY_THRESHOLD
```

```

    }
)

# Initialize the Cross-Encoder model used for re-ranking
model = HuggingFaceCrossEncoder(model_name=RERANKER_MODEL)

# Initialize the compressor which uses
# the Cross-Encoder model for re-ranking
compressor = CrossEncoderReranker(model=model, top_n=RETRIEVER_TOP_N)

# Create the ContextualCompressionRetriever which combines
# the compressor and retriever
compression_retriever = ContextualCompressionRetriever(
    base_compressor=compressor,
    base_retriever=retriever
)

```

The `HuggingFaceCrossEncoder` class is used to initialize a model capable of re-ranking documents. As previously discussed, a Cross-Encoder is used for ranking documents based on their relevance to a given query. Combined retriever and compressor can be used to perform highly accurate document retrieval tasks, where documents are not only retrieved but also optimized for relevance and context. The model used here is specified by the `RERANKER_MODEL`, as already mentioned, the `bge-reranker-v2-m3` was used during testing.

The `CrossEncoderReranker` class takes in the previously initialized Cross-Encoder model and the `top_n` parameter, which indicates the number of top retrieved documents that will be retrieved after re-ranking. The value of `RETRIEVER_TOP_N` was set to 2 during testing.

Finally, the pipeline constructs the `CrossEncoderReranker`, which combines both the retriever and the compressor. The purpose of this combined structure is to streamline the retrieval and ranking process. This two-step process improves the quality of the final results, ensuring that the documents returned are not only relevant but also ranked according to their true relevance to the user's query.

3.4.4 Prompt Template

A prompt system for language models is a structure by which input data is provided to a model in order to guide the model to generate a desired response. The prompt acts as a context or instruction that shapes how the language model processes the input and generates outputs. It can include specific instructions, examples, or questions, influencing

the model's behavior, output format, and the overall task it performs.

LangChain provides the `PromptTemplate` class, a core component used to define reusable prompt structures that can be filled with inputs. It allows to create templates for prompts, where placeholders are filled with variables or specific values during runtime.

For Qwen models the following format was employed:⁷

```
<|im_start|>system
You are Qwen, created by Alibaba Cloud. You are a helpful assistant.
You are a knowledgeable assistant specializing in answering
questions first aid procedures. Use the most relevant pieces of the
retrieved context below to support your answer.
If you don't know the answer or the context is insufficient,
state that you are unsure. Use a maximum of three sentences,
and keep your answer concise, accurate, and where applicable,
use medical terminology. <|im_end|>"" + ""
<|im_start|>user
Question: {question}
Context: {context}
<|im_end|>
<|im_start|>assistant<|im_end|>
```

For Phi models the following format was employed:⁸

```
<|system|>
You are an assistant for question-answering tasks about medical domain.
Use the following pieces of retrieved context to answer the question.
If you don't know the answer, just say that you don't know.
Use three sentences maximum and keep the answer concise.
<|end|>"" + ""
<|user|>
Question: {input}
Context: {context} <|end|>
<|assistant|>
```

⁷Following the prompt template: <https://ollama.com/library/qwen/blobs/1da0581fd4ce>

⁸Following the prompt template: https://github.com/microsoft/PhiCookBook/blob/main/md/02.QuickStart/Huggingface_QuickStart.md

3.4.5 Generation

After the `ContextualCompressionRetriever`, which combines both the retriever and the compressor, the last key components to add is part of generated responses.

As the first step is the initialization of a prompt template, which is determined by the function `process_prompt(args)`. This function selects an appropriate prompt structure based on the specific model type, as presented in the previous section.

Following the prompt definition, the script initializes the language model by executing the function `initalize(args)`. Once the model is returned, the corresponding tokenizer is loaded using the `AutoTokenizer.from_pretrained(model_id)` method. The tokenizer is a crucial component, as it converts text input into tokenized representations that can be processed by the model.

After initializing the tokenizer, the `AutoModelForCausalLM.from_pretrained(model_id, device_map=DEVICE_MAP)` loads the pre-trained language model. As in the retrieval phase, the `DEVICE_MAP` is set to `cuda` to accelerate processing.

```
# Initialize the prompt based on the model passed as an argument
prompt_template = process_prompt(args)

# Initialize the instructor model's ID
model_id = initalize(args)

# Load the tokenizer for the model
tokenizer = AutoTokenizer.from_pretrained(
    model_id,
    legacy=False,
    clean_up_tokenization_spaces=True
)

# Load the model for the model
model = AutoModelForCausalLM.from_pretrained(
    model_id,
    device_map=DEVICE_MAP
)

# Create a pipeline for generating model replies
# based on input context and question
pipe = pipeline(
    MODEL_REPLY,
```

```

        model=model,
        tokenizer=tokenizer,
        max_new_tokens=MAX_NEW_TOKENS,
        clean_up_tokenization_spaces=True,
        device_map=DEVICE_MAP
    )

    # Initialize a HuggingFace pipeline for local inference using
    # the created pipeline
    local_slm = HuggingFacePipeline(pipeline=pipe)

    # Create a chain for processing documents with the local model for
    # retrieval-augmented generation
    combine_docs_chain = create_stuff_documents_chain(
        local_slm,
        PromptTemplate(input_variables=["context", "question"],
            template=prompt_template)
    )

    # Create the full RAG chain
    rag_chain = create_retrieval_chain(
        compression_retriever,
        combine_docs_chain
    )

```

A text generation pipeline is then created using the Hugging Face `pipeline` function. This pipeline is responsible for managing the interaction between the model and tokenizer, ensuring that input queries are properly processed and responses are generated in a structured manner. The `MODEL_REPLY` is set to `\text-generation`, specifying that the model should generate text-based responses. `max_new_tokens` is the limit on the number of new tokens the model is allowed to generate, `MAX_NEW_TOKENS` during testing is set to 1024.

The constructed pipeline is then wrapped within a `HuggingFacePipeline` object, to ensure compatibility with `LangChain`.

The `create_stuff_documents_chain()` function subsequently constructs the document-processing chain using, integrating the language model `local_slm` ensuring that it can process retrieved documents and the prompt template `prompt_template` structured to accept two input variables `\context`, retrieved document content, and `\question`, the user's input query.

The final stage of the script involves creating the full RAG pipeline by integrating the retrieval mechanism with the language model. This is accomplished through the function `create_retrieval_chain()`, which takes: the `compression_retriever`, the retriever mechanism that selects relevant documents, and `combine_docs_chain`, the previously defined chain that formats the retrieved information and interacts with the language model.

Chapter 4

Evaluation

Evaluating a Retrieval-Augmented Generation (RAG) system is a crucial step. In this thesis, our RAG system has been designed to provide accurate responses in the domain of first aid, where information quality is critical. The objective of this chapter is to analyze the system's performance across different dimensions, focusing on the relevance of the retrieval phase, the quality of the generated responses, and the computational efficiency of the model.

First, Section 4.1 introduces the evaluation dataset used as the basis for all conducted tests. The selection of appropriate test data is essential, as a representative dataset ensures a reliable assessment of the system's performance.

Next, Section 4.2 focuses on evaluating the information retrieval phase. Since the effectiveness of this step directly impacts the quality of the generated responses, we analyze different document chunking sizes and how different size affects the retrieval of relevant context. Additionally, we compare the performance of two word embedding models introduced in Section 3.3.2 to determine which one optimizes context retrieval most effectively.

Once the best word embedding model and chunk size are selected, we proceed to the evaluation of the generation phase. Section 4.3 details the metrics used to assess the quality of the responses produced by the system.

Given the critical nature of the medical domain, ensuring that the generated responses are not only coherent and informative but also aligned with authoritative medical knowledge is essential. A human evaluation was conducted, presented in Section 4.3.1. Additionally, to further analyze the generated responses, an expert was consulted and asked to provide an evaluation, which is presented in Section 4.3.2. Human reviews play a key role in assessing response quality, particularly in sensitive domains like healthcare, where automated metrics may not fully capture the nuances of correctness.

For a RAG-based system to be practical response times must be within acceptable limits. Section 4.3.3 examines the latency of the models presented in Section 3.2.

4.1 Evaluation Dataset

An evaluation dataset is a fundamental component to assess the performance of a model. It is a curated set of data specifically designed to test the accuracy, reliability, and overall effectiveness of a system by comparing its outputs to a predefined standard. In the context of question-answering models, an evaluation dataset consists of a collection of questions alongside their corresponding correct answers, referred to as the ground truth.

For this thesis, 50 questions in the evaluation dataset were sourced from the National CPR Association’s free practice test. These tests are publicly available and are intended to help candidates prepare for various National CPR Association’s certification exams.¹

Originally, the questions extracted from the National CPR Association test were in multiple-choice format. However, given that the language model in this study is designed to handle open-ended question answering, the evaluation dataset was changed. Each multiple-choice question was reformulated as an open-ended question.

Furthermore, in the case of multiple-choice questions, the ground truth is typically the officially designated correct answer. However, in the context of open-ended questions, the ground truth often requires additional context to ensure clarity and completeness. To ensure the integrity of the ground truth in this transformation, additional context was incorporated where necessary. This included integrating explanatory details that clarify why a particular answer is correct. When the official test provided explanations for the correct answers, they were included verbatim. In cases where such explanations were missing, supplementary context was manually added based on the study material provided by the National CPR Association.² The goal was to create a more informative and comprehensive ground truth, ensuring that it provides not only the correct response but also an understanding of its rationale.

The table below illustrates an example of a multiple-choice question converted into an open-ended format, along with its corresponding ground truth.

Original Multiple-Choice Format:

¹All National CPR Association’s test are available: <https://www.nationalcprassociation.com/free-cpr-study-guide>

²Available at: <https://www.nationalcprassociation.com/free-cpr-study-guide>

Question	Correct Answer
<p>If performing rescue breathing, you should:</p> <ul style="list-style-type: none"> a) Avoid excessive ventilation b) Provide a breath every 2 seconds c) Do not allow the chest to expand between breaths d) All of the above are correct 	Avoid excessive ventilation

Reformulated Open-Ended Format:

Question	Ground Truth
When performing rescue breathing, what should be avoided to ensure effective ventilation?	Avoid excessive ventilation. Excessive ventilation may cause air to build up in the person’s stomach, which may lead to difficulty providing good compressions. Gastric distention may also lead to vomiting by the victim which can result in aspiration (inhalation of the vomited material) and pneumonia.

Further examples of reformulated questions are provided in appendix B.

4.2 Information Retrieval

To evaluate the retrieval performance of the proposed system, we follow the evaluation framework presented in [YGZ⁺24]. It specifies two key metrics to assess the retrieval component:

- **Relevance (Retrieved Documents to Query):** This metric quantifies the correlation of the retrieved documents with the information specified in the query. High relevance scores indicate that the retrieved documents contain information that is directly relevant to answering the query.
- **Accuracy (Retrieved Documents to Document Candidates):** This metric evaluates the correctness of the retrieved documents relative to a predefined set of candidate documents. It provides insight into the retrieval system’s ability to return the most relevant and accurate information from the available knowledge base, ensuring that the retrieved contexts are both factually and contextually appropriate.

As mentioned in the previous section, additional context was incorporated into the ground truth answers the majority of which directly from the knowledge base. Given the absence of document candidates in the evaluation set, the ground truth is used as document candidates.

As outlined in Section 3.3.2, two distinct embedding models, MedEmbed-small-v0.1 and bge-small-en-v1.5, were tested for the retrieval phase. The objective of this evaluation is to determine the ability of each model to retrieve relevant context from a subset of 30 questions. The tests were repeated three times with varying chunk sizes: 500, 1000, and 1500 tokens.

Table 4.1 below summarizes the number of instances where context retrieval was unsuccessful for each model across different chunk sizes.

Table 4.1: Number of questions with missing context out of 30

Chunk size (tokens)	MedEmbed-small-v0.1	bge-small-en-v1.5
500	3	4
1000	3	4
1500	3	3

As shown in Table 4.1, the results indicate that MedEmbed-small-v0.1 failed to retrieve context in only 3 cases, regardless of the chunk size. In contrast, bge-small-en-v1.5 was unable to retrieve relevant context in 4 instances when using chunk sizes of 500 and 1500 tokens. Notably, bge-small-en-v1.5 was only able to match MedEmbed-small-v0.1 in terms of successful context retrieval when using the larger chunk size of 1500 tokens. This suggests that the general-purpose word embeddings bge-small-en-v1.5 requires a larger chunk size to perform comparably to the fine-tuned MedEmbed-small-v0.1.

To further assess retrieval quality, the cosine similarity was calculated to measure the semantic closeness of the retrieved context to both the query and the ground truth. Cosine similarity is a metric that quantifies the cosine of the angle between two vectors in a vector space. In this case, the vectors represent the query and the retrieved documents. A cosine similarity score close to 1 indicates that the two vectors are very similar, meaning that the retrieved documents are highly relevant to the query, while a score closer to 0 suggests little to no similarity.

This evaluation was repeated with two larger versions of the embedding models: MedEmbed-large-v0.1³ and bge-large-en-v1.5⁴, which represent scaled-up versions of the models used

³Available at: <https://huggingface.co/abhinand/MedEmbed-large-v0.1>

⁴Available at: [:BAAI/bge-large-en-v1.5](https://huggingface.co/BAAI/bge-large-en-v1.5)

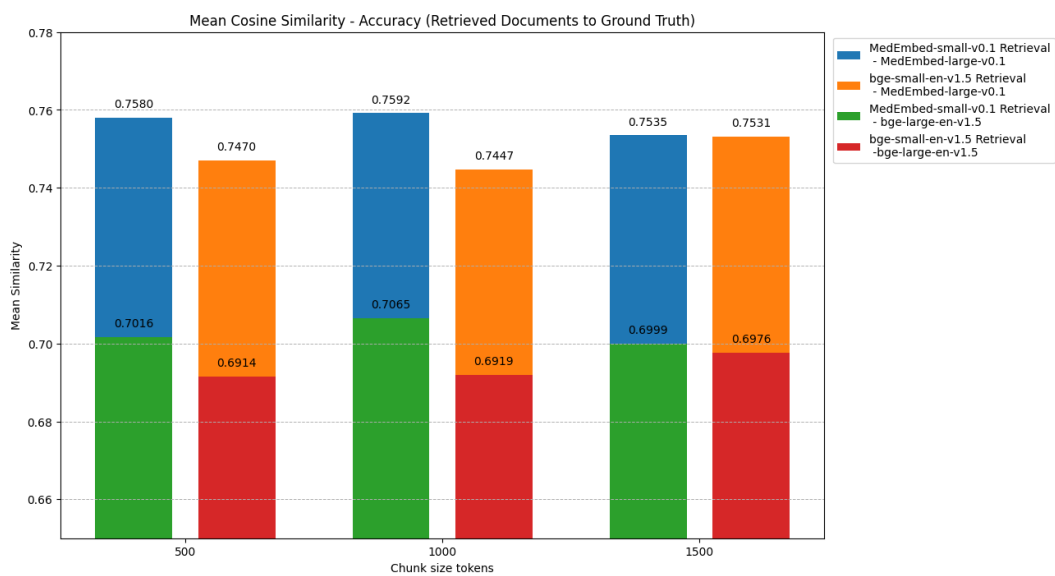


Figure 4.1: The mean of the cosine similarity for Accuracy (Retrieved Documents to Ground Truth). Blue and green bars illustrate the tests with the MedEmbed-small-v0.1 model. Blue is the mean cosine similarity computed with MedEmbed-large-v0.1. Green is the mean cosine similarity computed with bge-large-en-v1.5. Red and orange bars are the results with the bge-small-en-v1.5. Orange is the mean cosine similarity computed with MedEmbed-large-v0.1. Red is the mean cosine similarity computed with bge-large-en-v1.5.

in the retrieval phase. These larger models were chosen to ensure greater precision during evaluation. Both models were tested with the same three chunk sizes.

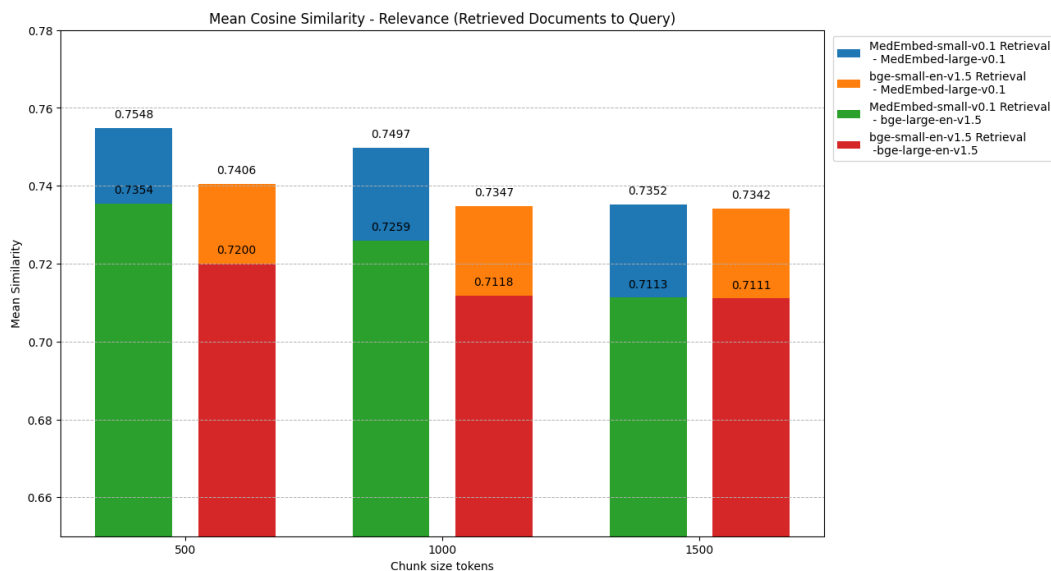


Figure 4.2: The mean of the cosine similarity for Relevance (Retrieved Documents to Query). Blue and green bars illustrate the tests with the MedEmbed-small-v0.1 model. Blue is the mean cosine similarity computed with MedEmbed-large-v0.1. Green is the mean cosine similarity computed with bge-large-en-v1.5. Red and orange bars are the results with the bge-small-en-v1.5. Orange is the mean cosine similarity computed with MedEmbed-large-v0.1. Red is the mean cosine similarity computed with bge-large-en-v1.5.

As illustrated in Figures 4.1 and 4.2, the cosine similarity values for both relevance and accuracy were significantly higher for chunk sizes of 500 and 1000 tokens when using MedEmbed-small-v0.1. At a chunk size of 1500 tokens, both models exhibited near-identical performance. The results for the 500 and 1000 token chunk sizes can be attributed to the higher frequency of retrieval failures, which lowered the overall mean cosine similarity for the bge-small-en-v1.5. Meanwhile, the values for 1500 token chunk size, suggests that, when context is successfully retrieved, the performance of both models converges.

Given that MedEmbed-small-v0.1 consistently retrieved relevant context more frequently, there is limited justification for continuing to use bge-small-en-v1.5. Therefore, MedEmbed-small-v0.1 will be used for all subsequent tests in this study.

Further analysis of the relationship between chunk size, accuracy, and relevance reveals interesting trends. As illustrated in Fig. 4.2, one prominent trend is the correlation between chunk size and relevance: as chunk size increases, relevance tends to decrease. Different factors could contribute to this:

- **Loss of Specificity:** Larger chunk sizes result in a loss of specificity, which could explain the decline in relevance. Smaller chunks are typically more focused on certain topics, ensuring that retrieved content aligns closely with the query, leading to higher cosine similarity scores. In contrast, as chunk size increases, a broader range of information is included, some of which may not directly relate to the query. Such information diminishes the relevance of the retrieved content. Consequently, even if a chunk contains both relevant and irrelevant information, its overall cosine similarity score is lower compared to smaller, more targeted chunks.
- **Query Sensitivity:** The nature of the query could further influence how chunk size impacts relevance. Specific queries tend to benefit from smaller, more focused chunks, as they are more likely to return content directly related to the query. In contrast, broader queries might benefit from larger chunks, which provide more contextual information. In the context of this thesis, as outlined in Section 4.1, queries are derived from multiple-choice questions reformulated into open-ended formats. As a result, these queries typically require concise, relevant context, meaning that larger chunks, which introduce added irrelevant information, tend to reduce retrieval accuracy.

Furthermore, as shown in Fig. 4.1, chunk size also plays a critical role in accuracy. The highest mean accuracy of retrieved context with MedEmbed-small-v0.1 model, occurs when the chunk size is set to 1000 tokens, suggesting that this size strikes an optimal balance by including enough relevant details to the ground truth while minimizing irrelevant information. Smaller chunks may lack content, which can negatively impact accuracy due to missing details. On the other hand, larger chunks, may include excessive irrelevant information, reducing cosine similarity between the context retrieved and the ground truth, diminishing accuracy.

Based on these findings, it is concluded that a chunk size of 1000 tokens is optimal and will be used for all subsequent tests in this thesis. Although this size may not maximize relevance, its high accuracy makes it the most suitable option, offering a reasonable trade-off between the two metrics.

4.3 Generation

After evaluating the retrieval phase, the next step is to assess the generation phase. Similarly to the previous section, this evaluation follows the metrics defined in the paper [YGZ⁺24]:

- **Generation Relevance (Generated Response to Query):** This metric evaluates how well the generated response aligns with the original query’s intent and content.

A high relevance score indicates that the response stays within the context of the user’s request, directly addressing the specific needs of the query.

- **Faithfulness (Generated Response to Retrieved Documents):** Faithfulness measures whether the generated response accurately reflects the information found in the retrieved documents. This metric ensures that the generated content remains consistent with the source material, minimizing the risk of hallucinations or inaccuracies.
- **Correctness (Generated Response to Ground Truth):** This metric compares the generated response with a reference response, assessing its factual and contextual accuracy.

As mentioned in the previous Section 4.2, the following configuration is used during testing:

- A chunk size of 1000 tokens with a 10% overlap during document retrieval.
- The MedEmbed-small-v0.1 embedding model is used in the retrieval process, from the creation of the vector dataset to the computation of the similarity score.

As noted in Sections 3.2.1 and 3.2.2, the models evaluated in this phase include Qwen2.5-0.5B-Instruct, Qwen2.5-1.5B-Instruct, Qwen2.5-3B-Instruct, Phi-3-mini-instruct, and Phi-3.5-mini-128k-instruct. Similarly to the previous section, to quantify the evaluation metrics, we use cosine similarity to compare vector representations of the elements. The MedEmbed-large-v0.1 embedding model is used to generate vector embeddings for the metrics discussed above. In contrast to the retrieval phase, which was conducted on a subset of the evaluation set, the generation evaluation is performed on the entire test set of 50 questions.

As shown in Figure 4.3, the model with the highest cosine similarity for generation relevance is Qwen2.5-3B-Instruct, indicating that it generates responses that closely align with the queries. The next top-performing models in terms of relevance are Phi-3-mini-instruct and Phi-3.5-mini-128k-instruct, which also show strong alignment with the queries. In contrast, Qwen2.5-0.5B-Instruct, and Qwen2.5-1.5B-Instruct rank lowest in relevance, suggesting that these smaller models struggle to maintain query alignment.

A deeper analysis of Qwen2.5-3B-Instruct reveals an interesting trend: some responses generated by this model contain repetition or even an additional question before addressing the submitted query. This added noise could artificially boost the generation relevance metric.

As illustrated in Figure 4.4, the faithfulness evaluation shows that Qwen2.5-0.5B-Instruct performs the best in maintaining consistency with the retrieved documents. Despite its

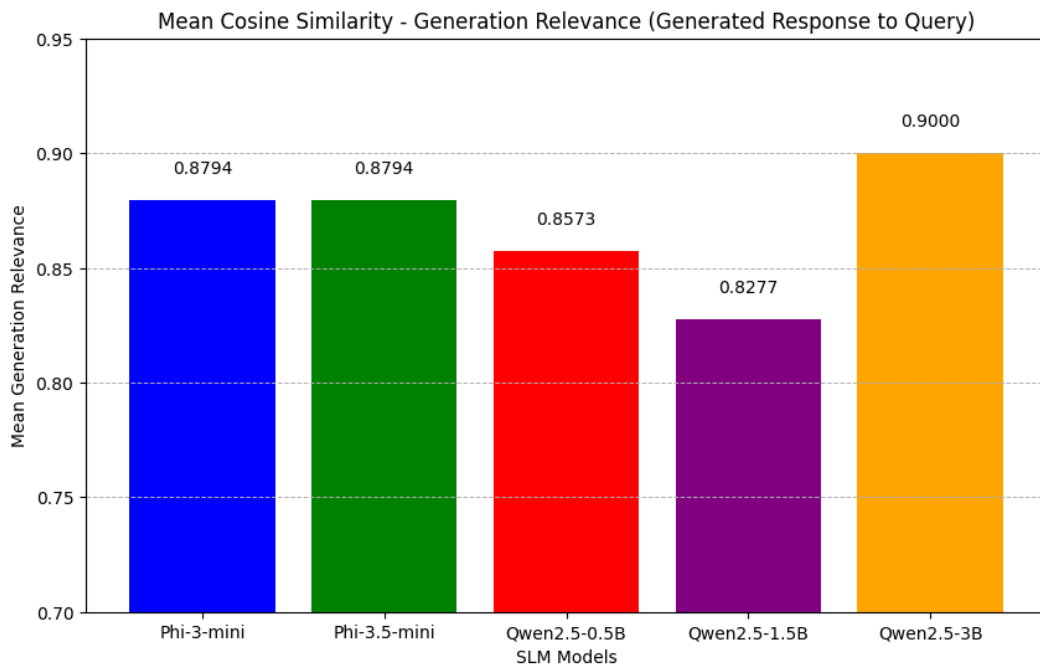


Figure 4.3: Comparison of Mean Generation Relevance Across Models.

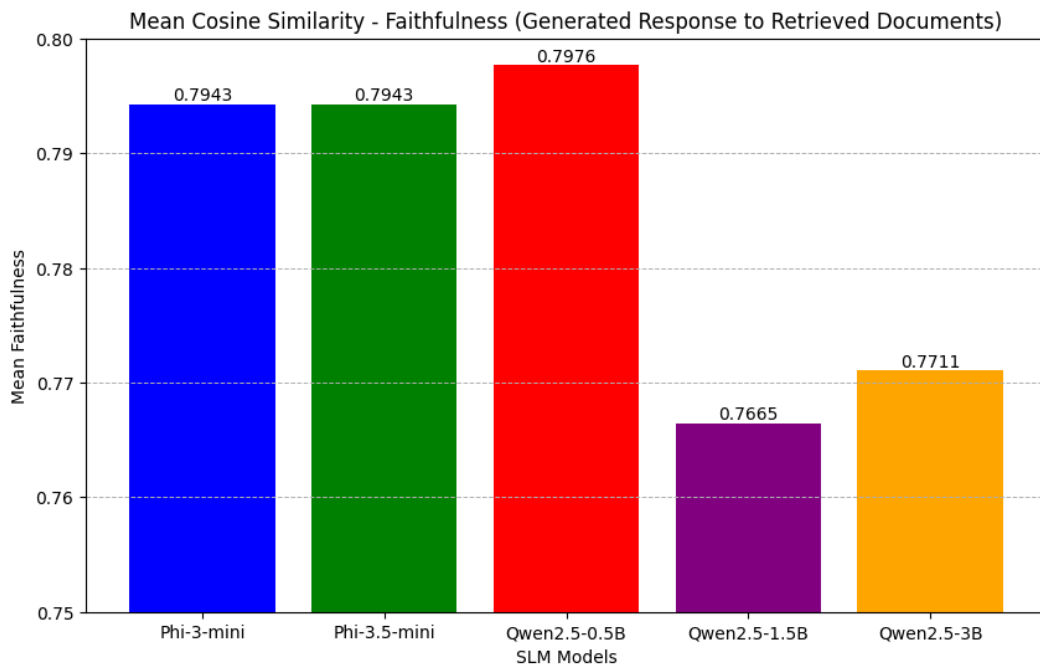


Figure 4.4: Comparison of Mean Faithfulness Across Models.

lower relevance score, such high faithfulness suggests that its generated responses remain faithful to the retrieved context. In second place, both Phi-3-mini-instruct and Phi-3.5-mini-128k-instruct demonstrate strong faithfulness, effectively extracting and utilizing information from the retrieved documents. Qwen2.5-3B-Instruct ranks lower in faithfulness, suggesting that while it produces relevant responses, it may introduce content that deviates from the retrieved documents. Finally, by ranking the lowest both in faithfulness and generational relevance, the results for Qwen2.5-1.5B-Instruct indicate a difficulty in generating responses that effectively address the query and accurately report information from the retrieved context.

Upon closer inspection, we observe that all Qwen models tend to repeat the retrieved context verbatim, which may explain the high faithfulness score for Qwen2.5-0.5B-Instruct, where it explicitly includes phrases such as “The text explicitly states:” or “Additionally, it states that:” before reporting pieces of context.

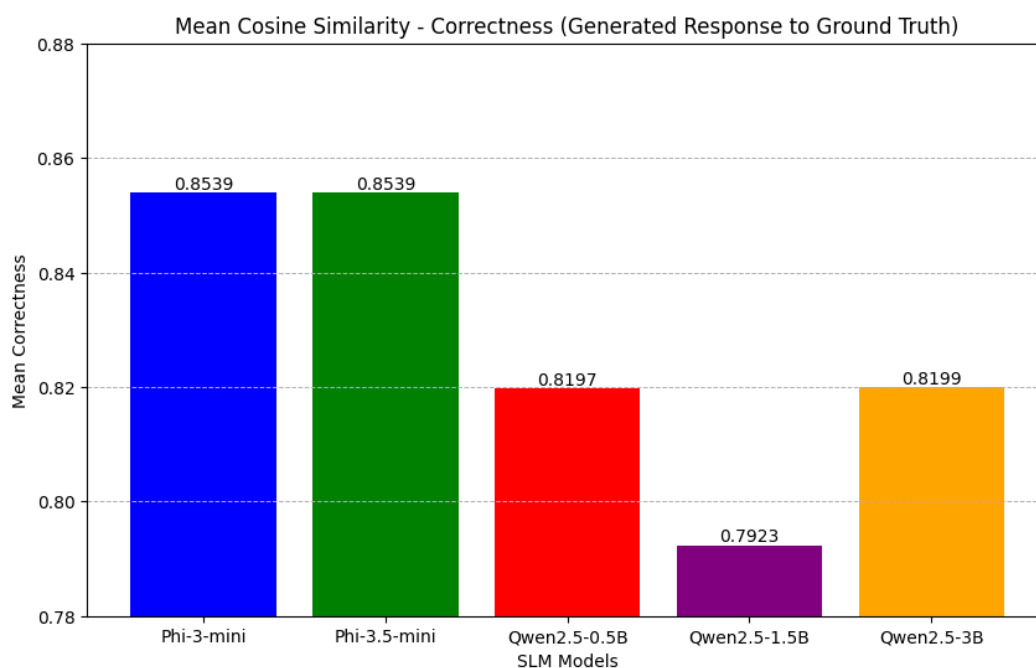


Figure 4.5: Comparison of Mean Correctness Across Models.

In the correctness evaluation, Phi-3-mini and Phi-3.5-mini achieve the highest scores, demonstrating their ability to generate responses that align with the reference ground truth. Qwen2.5-3B and Qwen2.5-0.5B achieve similar results, showing good performance in generating accurate responses, although behind the Phi models. Finally, Qwen2.5-1.5B ranks the lowest in correctness, further supporting its weaker performance across all evaluation metrics.

To further analyze the results in correctness, a subset of responses will be manually reviewed in the next section to determine if they align with the correctness metric.

4.3.1 Human Evaluation

The accuracy of generated responses is a critical factor, particularly in high-stakes domains such as first aid. The potential consequences of misinformation in this field emphasize the importance of having reliable methods to assess correctness. To further validate the metrics presented in the previous section, a manual assessment of the generated responses was conducted to determine whether the evaluation criteria were effectively capturing correctness. To ensure a comprehensive evaluation, the responses from the complete set of 50 questions were manually examined. The following criteria were applied to assess the accuracy of the responses:

- **Content:** The response must include the key information contained in the ground truth. If any crucial details were missing, the response was marked as incorrect.
- **Noise and Formatting Issues:** Responses containing elements such as additional generated questions or unrelated characters were not automatically deemed incorrect. This issue was particularly prevalent in responses generated by Qwen models.
- **Accuracy of Additional Information:** If the response contained additional information beyond the ground truth, it was assessed for correctness. If the additional content was factually accurate, the response was still considered correct.
- **Impact of Additional Information on Interpretability:** In cases where additional content was present, it was evaluated to determine whether it affected the interpretability of the response. If the extra details led to contradictions or details that could lead to misinterpretation, the response was marked as incorrect.

As illustrated in Figure 4.6, from these results, it is evident that the Phi models significantly outperform the Qwen models in terms of accuracy. The Phi-3-mini-128k-instruct and Phi-3.5-mini-instruct models achieved correctness rates of 86.00% and 80.00%, respectively, while the Qwen models exhibited lower performance, with correctness rates ranging between 36.00% and 62.00%. This indicates that Phi models are more reliable in generating responses that closely match the ground truth.

Another observation from the evaluation was the variation in response length across different models. Despite instructions to provide concise answers, the Qwen models consistently produced longer responses than the Phi models.

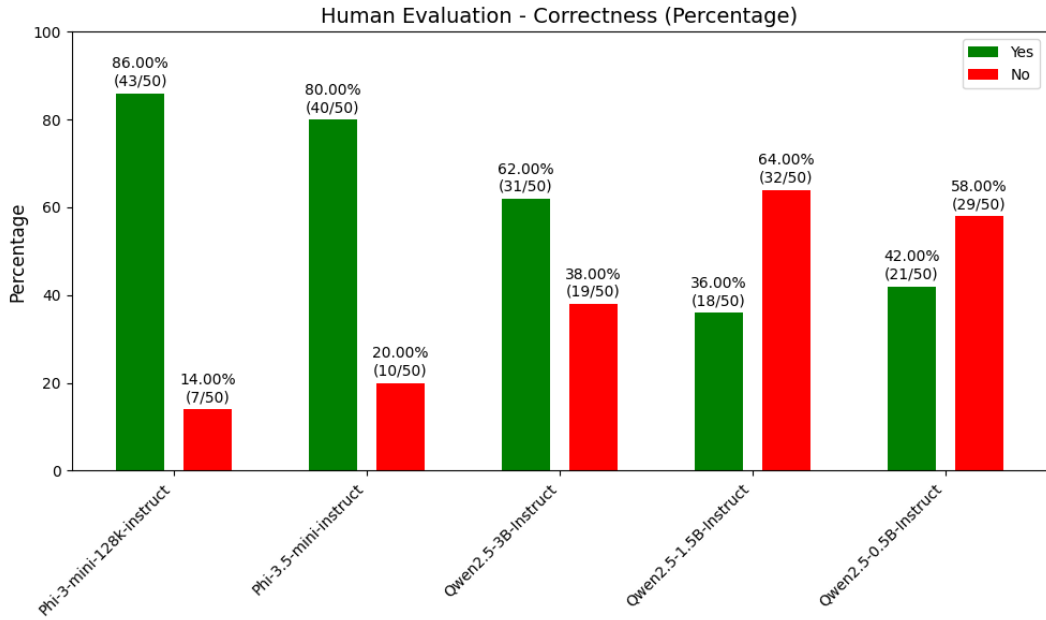


Figure 4.6: Results of manual evaluation.

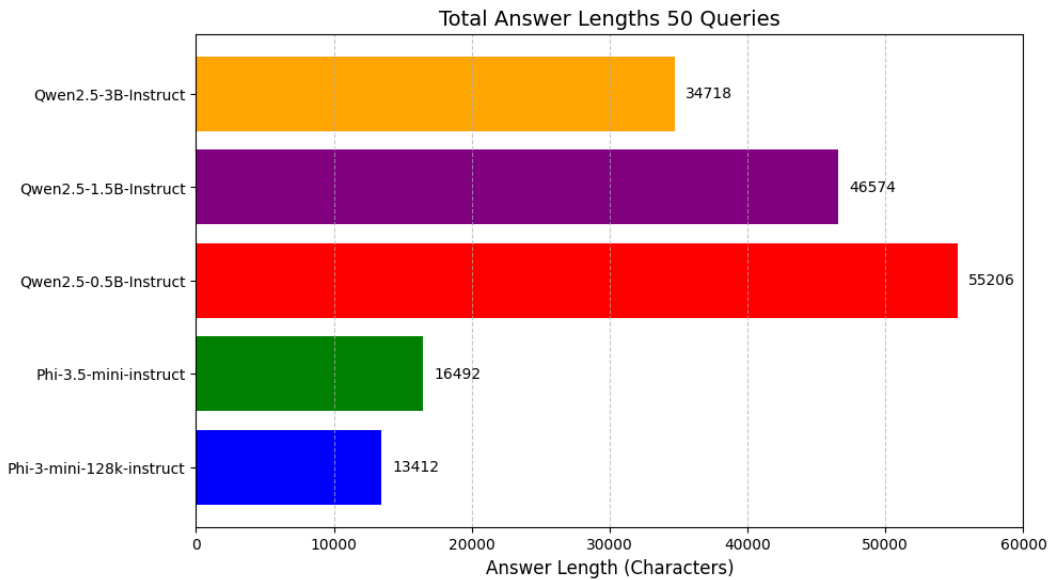


Figure 4.7: Total length of the generated responses of the evaluation set.

As shown in Figure 4.7, the analysis reveals that the Qwen models, particularly the smaller variants Qwen2.5-1.5B and Qwen2.5-0.5B, generated responses significantly longer than those produced by the Phi models. Upon closer examination, it was observed that the excessive length in Qwen responses is often attributed to:

- **Redundant Information:** Qwen responses frequently included repetitive statements or superfluous details that did not contribute to the answer.
- **Unnecessary Questions:** Many responses generated by the Qwen models included follow-up questions or speculative statements that were not required for answering the query.
- **Irrelevant Data:** Some responses contained additional information that, while somewhat related to the topic, was not directly pertinent to the question at hand.

Given the importance of accuracy and conciseness in the domain of first aid, the observed tendencies in Qwen models present significant concerns. Emergency situations require quick and precise information delivery, and the presence of redundant or misleading information could hinder the effectiveness of a first aid chatbot. The superior performance of the Phi models in both accuracy and brevity suggests that they are better suited for deployment in real-world first aid applications.

4.3.2 Expert Evaluation

Given the critical nature of first aid and emergency response, the most reliable method for evaluating a RAG system remains expert human judgment. Expert evaluation provides a nuanced analysis of the generated responses, ensuring that the system's output aligns with practical, real-world knowledge.

However, securing expert evaluation for this study proved to be challenging. Healthcare professionals, particularly those engaged in emergency medical services, operate under demanding schedules. Their expertise is invaluable, but their professional obligations often prevent them from participating in studies. Additionally, given the linguistic barrier, as the study was conducted in English, finding a healthcare expert proficient in both the language and first aid protocols introduced an additional layer of complexity.

After extensive research and outreach, an expert was identified who had the necessary qualifications to assess the system performance. The selected evaluator, Professor Andrea Gatto, is an English teacher with extensive experience in emergency medical response as a Milite in Public Assistance. The term Milite refers to an emergency rescuer responsible for direct intervention in rescue activities. These professionals are trained to provide immediate first aid, perform Basic Life Support (BLS) procedures, administer oxygen, and execute

life-saving maneuvers in trauma cases. Their expertise encompasses the assessment and stabilization of patients in a variety of emergency situations, including accidents, medical crises, and natural disasters [Wik25].

To conduct the evaluation, a structured survey was developed. This survey comprised ten questions, with two random questions selected for each first aid subject to ensure complete coverage of the knowledge. For each question, three responses were presented, corresponding to the three top-performing models identified in previous evaluations: Phi-3-mini-128k-instruct, Phi-3.5-mini-instruct, and Qwen2.5-3B-Instruct. The expert was asked to rate each response on a scale from 1 to 5. In addition to numerical scoring, the survey provided an opportunity for textual feedback, allowing the expert to elaborate on the reasoning behind each rating. This combination of quantitative and qualitative analysis ensured a more comprehensive assessment of model performance.

The results of the expert evaluation are visualized in Figure 4.8, which presents a heatmap summarizing the scores assigned to each model’s responses. The findings reveal a clear hierarchy in model performance. Among the three models, Phi-3.5-mini received the highest overall scores, closely followed by Phi-3-mini. Qwen2.5-3B-Instruct, in contrast, was consistently rated lower.

The textual feedback provided by the expert offers valuable insights for each model:

- **Qwen2.5-3B-Instruct:** The responses generated by this model were frequently described as redundant and excessively verbose. The expert noted that the model introduced unnecessary follow-up questions, which, while informative in a general setting, were deemed ineffective in emergency scenarios where time is critical. In one instance, the verbosity was considered beneficial, as it provided a more comprehensive explanation. However, even in this case, the expert reiterated that, in an emergency context, brevity and clarity are paramount.
- **Phi-3-mini-128k-instruct:** The responses generated by Phi-3-mini-128k-instruct were consistently described as concise and direct. In several cases, the expert found its output to be nearly equivalent to that of Phi-3.5-mini-instruct. However, one notable critique was that while the content was correct, the presentation of information could be improved. Specifically, better formatting could be used to emphasize the priority of actions in emergency situations, ensuring that responders could immediately identify the most urgent steps to take.
- **Phi-3.5-mini-instruct:** Like Phi-3-mini-128k-instruct, the responses from Phi-3.5-mini-instruct were regarded as concise and direct. Furthermore, they were also described as clearer and more complete, with additional information. This distinction highlighted the advantage of Phi-3.5-mini-instruct over Phi-3-mini-128k-instruct. In the specific case where Phi-3-mini-128k-instruct’s response was critiqued for format-

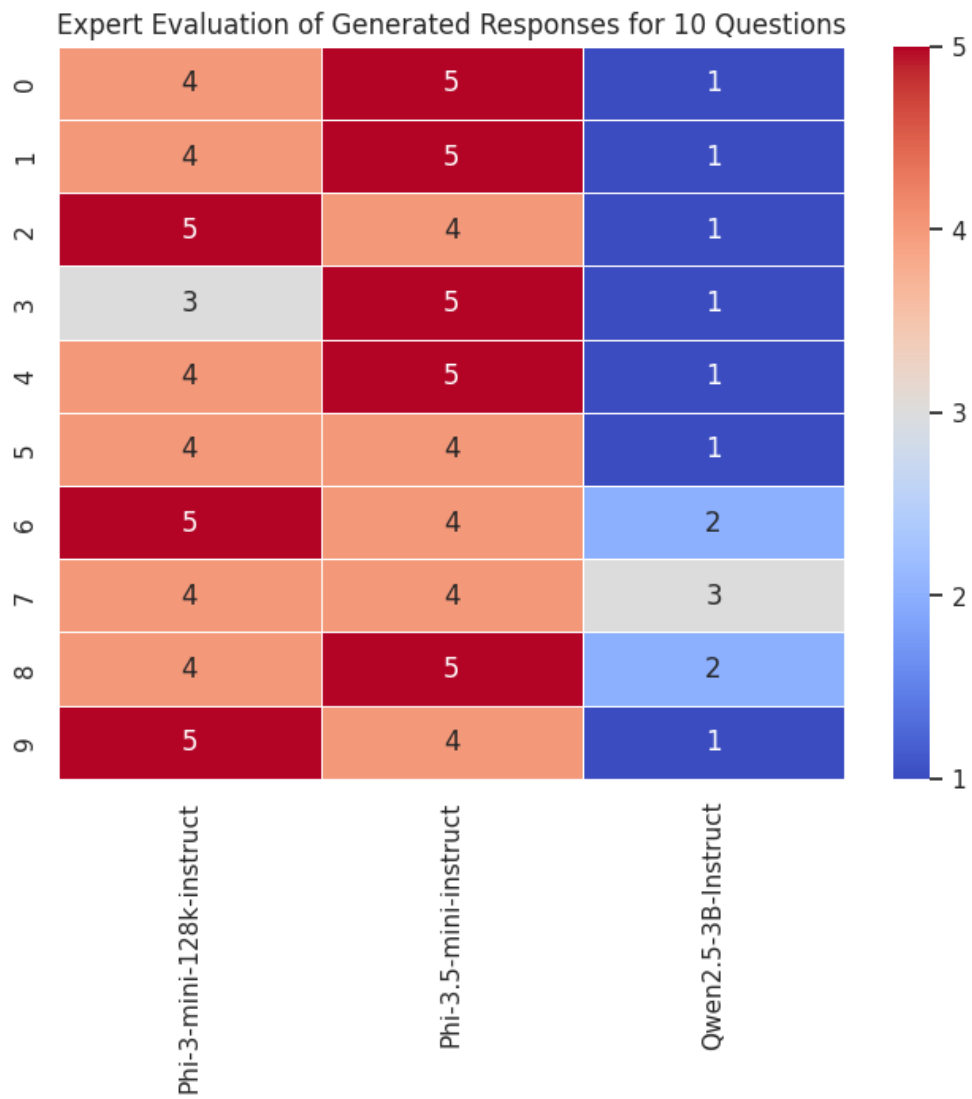


Figure 4.8: Evaluation of the generated response for each model.

ting, Phi-3.5-mini-instruct’s response was praised for effectively prioritizing actions in a more structured manner.

At the end of the survey, the expert was asked to provide an overall assessment of the potential utility of such a tool in real-world applications. Professor Gatto emphasized that in any emergency situation, the first and most crucial action should always be to call emergency services. The tool, while potentially useful, should only be considered as a last resort. Additionally, an alternative use case was proposed: the expert suggested that the system could serve as an educational resource. By directing users to source materials and providing structured guidance, the chatbot could aid in first aid training, allowing users to familiarize themselves with emergency procedures before encountering real-life situations.

In conclusion, the expert evaluation reaffirmed the importance of precision, clarity, and structured prioritization in emergency response communication. While Phi-3.5-mini-instruct demonstrated the highest overall performance, Phi-3-mini-128k-instruct remained a strong contender. Qwen2.5-3B-Instruct, despite its detailed responses, was found to be less suitable for emergency scenarios due to its verbosity.

4.3.3 Latency

This section focuses on latency, specifically “Single Query Latency” as defined in the paper [YGZ⁺24]. Latency refers to the time required for a model to process a single input query and generate a corresponding output. This metric is especially important for real-time applications that demand rapid response times, such as emergency situations requiring first aid.

Single Query Latency in the context of Retrieval Augmented Generation, as described in [YGZ⁺24], is defined as the duration from the moment an input is provided to a model until the output is generated, including both retrieval and generation phase. As outlined in Section 3.1.2, all latency tests were conducted on the NVIDIA Jetson AGX Xavier platform. The testing environment utilized CUDA 11.4, constrained by the JetPack 5.1.4 SDK. CUDA libraries enables GPU acceleration, optimizing inference.

The box plot shown in Figure 4.9 illustrates latency results for all 50 queries of the evaluation set. A box plot graphically represents the distribution of a dataset based on five key summary statistics: minimum, first quartile (Q1), median, third quartile (Q3), and maximum. The interquartile range (IQR), the difference between Q3 and Q1, represents the middle 50% of the data. Whiskers extend from the quartiles to the smallest and largest values within 1.5 times the IQR, with any values outside this range considered outliers, represented by circles. A smaller IQR indicates more consistent performance, while a larger IQR suggests greater variability. Outliers indicate extreme latency values [Str24].

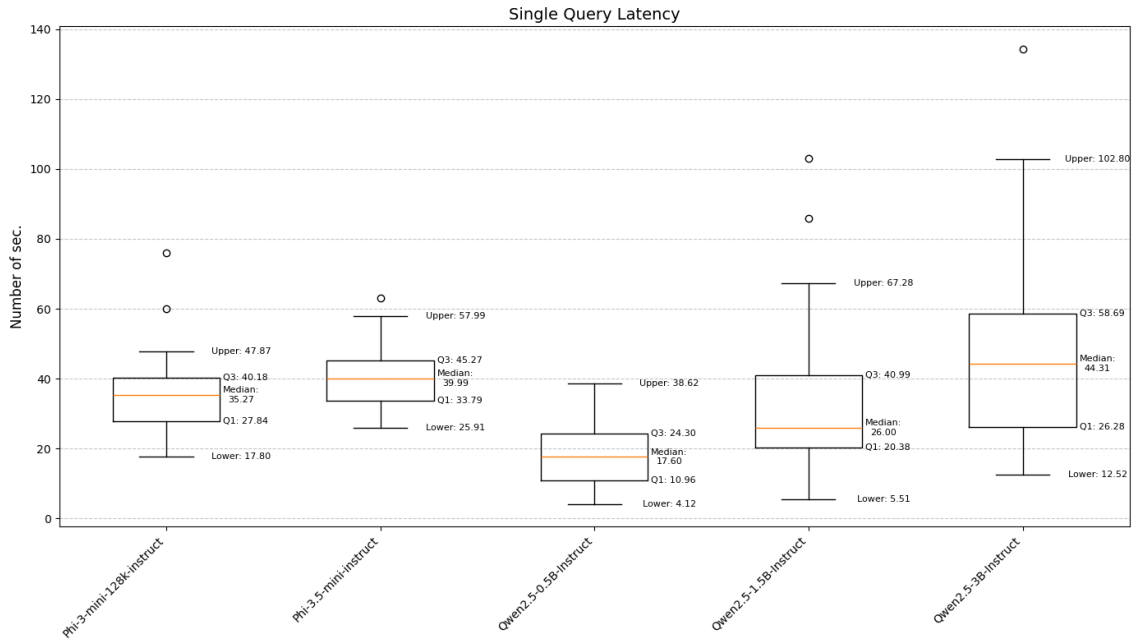


Figure 4.9: Box and whisker plot of Single Query Latency of the generated responses of the evaluation set.

The measured latencies show that smaller models, such as Qwen2.5-0.5B-Instruct, exhibit lower median latency values compared to larger models, such as Qwen2.5-3B-Instruct, which demonstrate higher processing times and a wider distribution of latency values.

A notable observation is the variation in the interquartile range (IQR) across models. As stated above, a larger IQR indicates more latency variability, which could impact real-time responsiveness. For example, Qwen2.5-3B-Instruct shows the highest latency spread, with outliers extending to 134.28 sec. This significant variance suggests that, while the model may achieve low latencies in some cases, it is prone to occasional high latencies, which could be problematic for time-sensitive applications. On the other hand, Qwen2.5-0.5B-Instruct exhibits the lowest median latency and a narrow IQR, indicating more stable and predictable performance. This makes it more suitable for edge computing scenarios where consistent response times are crucial. However, given the first aid context of this thesis and the results from the previous section, the unreliable correctness of responses from this model renders it unsuitable for this use case.

Interestingly, Phi-3-mini-128k-instruct and Phi-3.5-mini-instruct outperform Qwen2.5-3B-Instruct, despite being larger models, having both 3.8b parameters. These models also display smaller IQR values and a narrower spread between the lower and upper whiskers, indicating more stable latency performance compared to Qwen2.5-3B-Instruct and Qwen2.5-1.5B-Instruct. This suggests that these models are optimized for better computational

efficiency and exhibit lower variance in response times, making them more reliable for real-time applications.

It should be noted that these results were obtained using base models without optimizations such as quantization. Applying quantization, which reduces the precision of model parameters to enhance computational efficiency, could potentially improve performance and reduce latency.

Chapter 5

Other Use Cases

The following sections present two additional examples of how Retrieval-Augmented Generation (RAG) can be applied within the healthcare domain. These examples are intended as conceptual demonstrations rather than fully developed implementations.

It should be noted that the same RAG system used in the first aid use case was employed for these examples. Therefore, further customization of the system to meet the specific requirements of each task could lead to improved outcomes. The intent is to showcase the potential of RAG in the healthcare domain, with the understanding that tailored configurations and optimizations would be necessary to achieve optimal results for each application.

Furthermore, in the case study related to patient history in Section 5.1, no formal evaluation of the generated responses was performed, as the necessary medical expertise to accurately assess clinical recommendations was not available. The discussion will focus on observations regarding the system's output.

5.1 Patient History

Medical records often contain extensive and detailed information regarding a patient's health history, physical examinations, diagnoses, and treatments. Due to the complexity and volume of these records, a potential application of RAG in this domain could involve developing a tool to assist physicians by highlighting the most critical passages that require careful review. By summarizing key information and emphasizing important medical findings, such a tool could enhance the efficiency of medical professionals in reviewing patient histories and identifying relevant details more quickly.

Additionally, a RAG-based system could be used to generate preliminary diagnostic suggestions based on the extracted information. This could support the clinical workflow by providing physicians with an initial assessment of the case, potentially reducing the time required for decision-making.

However, it is important to note that such a tool should not replace human medical judgment but rather serve as an assistive technology that complements the expertise of health-care professionals. Ensuring human oversight remains central to the process is essential, as medical decision-making involves complex considerations that cannot be fully captured by automated systems.

To explore this concept, an experiment was conducted using an example medical record. For the knowledge base in retrieval, the Oxford American Handbook of Clinical Examination and Practical Skills ¹ was used. This reference provides comprehensive guidance on clinical examination techniques, interpretation of findings, and practical skills relevant to various medical scenarios. It serves as a valuable resource for clinicians by detailing best practices for patient assessment and diagnostic reasoning.

In this experiment, the RAG system was tested using the Qwen2.5-0.5B-Instruct model. The system generated the following response based on an example of medical record:

Generated Response
“Based on the provided case study, the patient presented with persistent chest pain and dyspnea on exertion, suggestive of pleural effusion or a more serious underlying condition like acute coronary syndrome. Given their history of smoking, they likely have a higher risk of developing cardiovascular issues. The presence of bilateral breath sounds without murmurs or other abnormalities further supports the possibility of pleural effusion rather than an acute cardiac event. Additionally, the absence of cyanosis and the normal neurological examination rules out any immediate cardiac problems. The presentation of angina and the presence of pleural effusion or pulmonary embolism require prompt evaluation. Therefore, it would be prudent to conduct further diagnostic tests such as a chest X-ray and possibly a transthoracic echocardiogram to confirm the diagnosis and assess the extent of the pleural problem.”

¹Available at : https://www.academia.edu/40908514/Oxford_American_Handbook_of_Clinical_Examination_and_Practical_Skills

Although the necessary medical expertise to formally evaluate the accuracy of this response was not available, certain aspects of the generated output were noteworthy. The model successfully identified key medical details, such as the patient’s history of smoking, a relevant risk factor for cardiovascular conditions. It also directed attention to critical clinical concerns, noting that “the presentation of angina and the presence of pleural effusion or pulmonary embolism require prompt evaluation.” Additionally, reasonable diagnostic recommendations were made, including the suggestion to conduct a chest X-ray and a transthoracic echocardiogram to confirm the suspected condition.

While this preliminary analysis indicates that a RAG-based system could provide meaningful insights when analyzing patient histories, it is crucial to emphasize that such a tool must be carefully validated before deployment in real-world clinical settings. Future work should involve collaboration with medical professionals to rigorously assess the system’s outputs, refine its retrieval and generation mechanisms, and ensure alignment with established medical guidelines and best practices.

5.2 Biometric Data

The RAG system was also tested for question-answering based on biometric data collected from patients. The dataset, provided by Prof. Delzanno, Prof. Mascetti, and Ph.D. Anna Khristodulo, included biometric measurements from patients with diabetes. The data encompassed various physiological parameters, including:

- **Heart Rate:** Number of heartbeats per minute over the dataset period.
- **Breath Frequency:** Number of breaths per minute.
- **Insulin Administration:** Total insulin administered during the monitoring period.
- **Blood Glucose Levels (Glicemia):** Concentration of glucose in the blood.
- **Physical Activity (Step Count):** Number of steps taken during the monitoring period.
- **Movement Index:** Movement intensity during the monitoring period.
- **Stress Index:** Stress levels.
- **HRV Score:** Heart Rate Variability (HRV) score reflecting the variation in time intervals between heartbeats.
- **Basal Insulin Administration:** Basal insulin administered.

For this experiment, the knowledge base was exclusively based on the provided biometric data.

During system prompting, additional contextual information was given regarding the files:

Additional Context Added to The System Prompt
Data for patient №3, collected from Comftech wearable devices and the Gaslini Medtronic platform: Comftech wearable device data: - hr3.csv - Heart rate - n_steps3.csv - Number of steps - bf3.csv - Breath frequency - si3.csv - Stress index - hrv3.csv - Heart rate variability - mi3.csv - Movement index Gaslini Medtronic platform data: - basal3.csv - Insulin basal data - correctcgm3.csv - Glycemic data - bolustest3.csv - Administered insulin The glycemic values were categorized as follows: - For a healthy person: - Low level: ≤ 70 - Normal level: 71 - 99 - High level: >99 - For a diabetic patient: - Low level: ≤ 69 - Normal level: 70 - 180 - High level: >181

The retrieval mechanism was designed to return relevant data for answering queries. Initially, data was formatted as CSV, but this approach resulted in poor retrieval performance. Switching to a more descriptive format, where each measurement was stated in natural language, e.g. “Heart Rate Max Heart Rate 2024-09-09: 232.0, Min Heart Rate 2024-09-09: 42.0, Mean Heart Rate 024-09-09: 90.37208218293951”, improved the retrieval system’s accuracy.

Despite these improvements, smaller models such as Qwen2.5-0.5B-Instruct struggled to correctly answer queries. Further testing with Phi-3-mini-128k-instruct led to significantly better results. An evaluation set of eight questions was constructed, following this format:

Question	Correct Answer
“What was the maximum heart rate on 2024-09-09?”	232.0

The model correctly answered six out of eight questions.

As stated in the previous section, this study should be seen as a conceptual demonstration rather than a final solution. A RAG system tailored specifically for biometric data analysis could be expected to provide more accurate and reliable responses.

Chapter 6

Conclusions and Future Work

This thesis explores the development of a first aid chatbot based on a Small Language Model (SLM) integrated with Retrieval-Augmented Generation (RAG). The system was designed for deployment on a Jetson Xavier AGX, an edge device equipped with a GPU, to accelerate response time and ensure accessibility in scenarios where internet connectivity may be unavailable.

Given the limitations and constraints imposed by the hardware, ChromaDB was utilized for vector storage of the knowledge base, while LangChain was employed for retrieval and pipeline mechanisms. The system dynamically accessed domain-specific knowledge from reliable sources, enabling it to generate informed responses, enhance reliability, and reduce hallucinations.

Expert evaluation results demonstrated that smaller models, when used within this framework, can achieve satisfactory performance, while maintaining a low model size and minimal computational requirements.

Despite these promising results, several challenges remain. The system's output is highly dependent on the quality of the retrieved context from the knowledge base. Additionally, the lack of memory retention prevents multi-step questioning, limiting its ability to support follow-up queries.

Several potential enhancements could be explored to improve the system:

- **Multi-step Question Handling:** As stated previously, the current system does not retain conversation history, making it unable to handle follow-up questions. Implementing a memory mechanism to track previous interactions would allow users to ask multi-step questions and receive responses that consider prior context.
- **Model Optimization:** This thesis test the use of base models; however, techniques

such as quantization and knowledge distillation could further reduce latency while preserving accuracy.

- **Improving Response Generation:** Fine-tuning the SLM could enhance response quality by refining parameterized knowledge while maintaining contextual augmentation. The use of synthetic aid dialogues could also contribute to more accurate answer generation.
- **Word Embedding Model:** In addition to fine-tuning the language model, improving the retrieval phase by employing a word embedding model specifically fine-tuned on first aid data could be beneficial. While the MedEmbed-small-v0.1 model used in this work is generally focused on healthcare data, additional fine-tuning on first aid content could enhance performance.
- **Knowledge Graph Indexing:** Introducing a hierarchical structure to document indexing could improve retrieval efficiency. A knowledge graph consists of nodes representing paragraphs and edges indicating semantic and lexical relationships. Given the multi-document nature of the knowledge base, this approach could be particularly useful.
- **Context Curation:** While reranking techniques were applied in this thesis, an additional post-retrieval processing step could involve context compression. Excessive context can introduce noise, whereas overly concise contexts may lack essential information. A selective compression strategy could help balance relevance and conciseness.

Appendix A

Knowledge Base

CPR – Infant casualty

1. Perform a scene survey.
2. Assess responsiveness. Gently tap the baby's feet.
3. If there is no response, send or call for medical help and an AED if available.

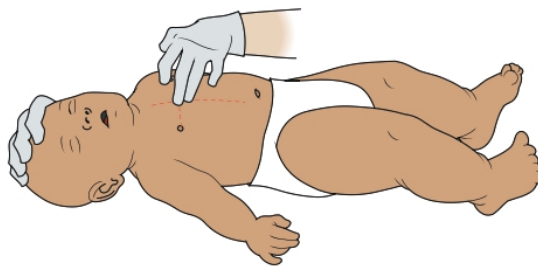
If you are alone with no phone perform 5 cycles of CPR (two minutes) then go for medical help. Carry the infant with you if possible.

4

4. Perform a Primary Survey
 - Open the airway.
 - Check for breathing for at least 5 and no more 10 seconds.



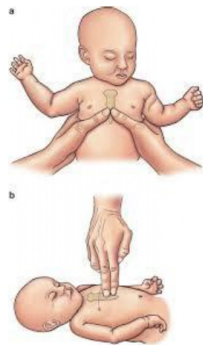
5. If the baby is not breathing, or not breathing effectively (agonal breaths) begin CPR



- Place two fingers on the breastbone just below the nipple line. Push down on the breastbone 1/3 the depth of the chest or about 4 cm (1 1/2 inches).

Figure A.1: Example of page of the St.John Ambulance Manual on CPR on infant casualty.

- **Compression delivery technique- 2 finger technique:** when one rescuer is present, the chest is compressed using two fingers on the lower half of the sternum, avoiding the xiphoid process (the very end of the sternum where it narrows and comes to a point). To landmark, place 2 fingers in the center of the infant's chest, just below the nipple line. Push down on the infant's chest one-third the depth of the chest, or approximately 1 ½ inches. Allow the chest to fully recoil (return to its neutral position) in between compressions. Compressions should be delivered at a rate of 100-120 compressions/minute.
- **2 thumb-encircling hand technique:** When two rescuers are present, the compression: ventilation ratio drops to 15:2, the same as for children. To perform this technique, position yourself at the infant's feet. Place your thumbs side by side on the center of the infant's chest just below the nipple line. Encircle the infant's chest so that the fingers of both hands support the infant's back. Use your thumbs to deliver compressions at the appropriate depth and rate (100-120 compressions/minute; 1/3 of the depth of the chest or approximately 1 ½ inches). This position allows another rescuer to support airway and breathing without getting in the way of the rescuer performing chest compressions, and is therefore the preferred technique when two rescuers are present. This technique also allows for more consistent chest compressions and superior blood flow and blood pressure compared to the 2-finger technique.



- **Compression: ventilation ratio-** The compression: ventilation ratio for 1 rescuer CPR in infants is 30:2, the same as for two rescuer CPR children.
- **When to call EMS-** it is far more common for infants to experience a respiratory arrest prior to going into cardiac arrest, unless the infant has a congenital heart problem; in

Figure A.2: Example of page of the National CPR Association study material on CPR on infant casualty.

Baby resuscitation:

Danger

Make sure you are safe to help, **do not put yourself in danger**

Response

- Gently tap the baby's shoulders and shout to try to wake the baby
- If the baby does not respond shout for help, but don't leave the baby yet

Airway

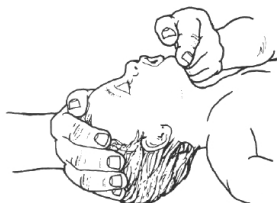
Carefully ensure the airway is open by gently tipping the baby's head back with the 'chin lift':

- Place your hand on the baby's forehead and gently tilt their head back. **DO NOT OVER-EXTEND THE BABY'S NECK**
- Using your fingertips under the point of the chin, lift the chin to open the airway

Breathing

Ensuring the airway is kept open; look, listen and feel to ascertain whether the baby is breathing normally. Do not take more than ten seconds to do this.

- If the baby is able to breath **normally**, consider the baby's injuries and place them in the recovery position (*see page 12*)



If the baby is not breathing normally:

- If you are not alone, ask the nearest person to **dial 999 for an ambulance immediately**. If you are alone and need to leave the baby to ensure help is on the way, perform resuscitation for about 1 minute first:
- Maintain the baby's airway by tilting their head and lifting the chin ensuring you do not over-extend the baby's neck
- Seal your mouth around the baby's nose and mouth
- Give 5 initial rescue breaths. Blow in just enough air to make the baby's chest rise visibly. **Be careful not to over inflate the baby's lungs**

Combine rescue breaths with chest compressions:

- Use 2 fingers to depress the baby's chest to a third of its depth
- Give 30 chest compressions at a rate of 100 compressions per minute
- Open the airway again by tilting the baby's head and lifting the chin, and give 2 more rescue breaths
- **Continue to administer 30 chest compressions followed by 2 rescue breaths**

Figure A.3: Example of page of the Actual First Aid notes on CPR on infant casualty.

Appendix B

Evaluation

1. The correct sequence for basic life support is:

- Airway, Breathing, Compressions
- Breathing, Airway, Compression
- Compressions, Airway, Breathing
- Breathing, Compressions, Airway

Incorrect
The 2010 American Heart Association (AHA) sequence was changed from the ABC's to CAB. By decreasing the delay in starting compressions, experts believe the survival rates will increase.

Figure B.1: Example of question with explanation of correct answer.

Table B.1: Example of open-question with ground truth.

Question	Ground Truth
What's the correct sequence for basic life support?	Compressions, Airway, Breathing. The 2010 American Heart Association (AHA) sequence was changed from the ABC's to CAB. By decreasing the delay in starting compressions, experts believe the survival rates will increase.

1. If an infant victim has a pulse but is not breathing, you may decide to do rescue breathing only. For this infant victim, you should provide a breath:

Every 1-2 seconds
 Every 3-5 seconds
 Every 5-6 seconds
 Every 10 seconds

Incorrect
Every 3-5 seconds

Figure B.2: Example of question without explanation of correct answer.

Table B.2: Example of open-question with ground truth with added context.

Question	Ground Truth
If an infant victim has a pulse but is not breathing, how often would you need to provide a breath when performing rescue breathing?	If the victim has a pulse but is not breathing, provide rescue breathing by providing 1 breath every 3 to 5 seconds (12-20 breaths per minute). Check for a pulse every 2 minutes- if at any time the victim loses their pulse, start chest compressions along with ventilations at a rate of 30:2.

What are the symptoms of shock? *

- **Risposta 1:**
Symptoms of shock include pale or blue-grey skin, cold and clammy skin, rapid and shallow breathing, anxiety, rapid pulse, dizziness, weakness, thirst, confusion, and changes in consciousness.
- **Risposta 2:**
Symptoms of shock include pale or blue-grey skin, rapid and irregular breathing, confusion, changes in consciousness, rapid and weak pulse, and anxiety. The individual may also experience extreme thirst, dizziness, and cold, clammy skin. These symptoms indicate inadequate blood flow to the body's organs due to blood loss or fluid loss.
- **Risposta 3:**
Based on the context provided, the symptoms of shock include: - Drowsiness - Confusion - Loss of consciousness - Extreme thirst - Pale skin color - Rapid heart rate - Cool, clammy skin - Rapid breathing - Anxiety These symptoms align with those described in the medical context, which also mentions signs like pale skin, cold and clammy skin, weak and rapid pulse, changes in level of consciousness, disorientation, confusion, dizziness, and fear.

	1	2	3	4	5
Risposta 1	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Risposta 2	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Risposta 3	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Note

La tua risposta _____

Figure B.3: Example of survey question for expert evaluation.

Bibliography

- [Adi24] AdinaTru. Discover the new multi-lingual, high-quality phi-3.5 slms, Aug 2024. URL: <https://techcommunity.microsoft.com/blog/azure-ai-services-blog/discover-the-new-multi-lingual-high-quality-phi-3-5-slms/4225280>.
- [Alt19] Rikka Altland. Nvidia unveils the jetson nano, a low-cost computer for experimenting with machine learning, Mar 2019. URL: <https://9to5toys.com/2019/03/18/nvidia-jetson-nano/>.
- [Ass19] Mike Assumma. Nvidia jetson nano: What is it, and what can it do?, Jun 2019. URL: <https://www.newegg.com/insider/nvidia-jetson-nano-dev-kit-makers-overview-what-can-it-do/>.
- [AVS24] Kamal Acharya, Alvaro Velasquez, and Houbing Herbert Song. A Survey on Symbolic Knowledge Distillation of Large Language Models, July 2024. arXiv:2408.10210. URL: <http://arxiv.org/abs/2408.10210>, doi: 10.48550/arXiv.2408.10210.
- [Bal24] Abhinand Balachandran. Medembed: Fine-tuned embedding models for medical / clinical ir, 2024. URL: <https://huggingface.co/blog/abhinand/medembed-finetuned-embedding-models-for-medical-ir>.
- [BBC⁺23] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, and Jianxin Ma. Qwen technical report, Sep 2023. URL: <https://arxiv.org/abs/2309.16609>, doi:10.48550/arXiv.2309.16609.
- [BGMMS21] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, FAccT '21*, pages 610–623, New York, NY, USA,

- March 2021. Association for Computing Machinery. URL: <https://dl.acm.org/doi/10.1145/3442188.3445922>, doi:10.1145/3442188.3445922.
- [Bha24] Pratik Bhavsar. Mastering rag: How to select a reranking model - galileo ai, 2024. URL: <https://www.galileo.ai/blog/mastering-rag-how-to-select-a-reranking-model>.
- [Bil24] Misha Bilenko. Introducing phi-3: Redefining what’s possible with slms, Apr 2024. URL: <https://azure.microsoft.com/en-us/blog/introducing-phi-3-redefining-whats-possible-with-slms/>.
- [BJCG21] Stephan Patrick Baller, Anshul Jindal, Mohak Chadha, and Michael Gerndt. DeepEdgeBench: Benchmarking Deep Neural Networks on Edge Devices, August 2021. arXiv:2108.09457. URL: <http://arxiv.org/abs/2108.09457>, doi:10.48550/arXiv.2108.09457.
- [BKDN23] Aptin Babaei, Parham M. Kebria, Mohsen Moradi Dalvand, and Saeid Navahandi. A Review of Machine Learning-based Security in Cloud Computing, September 2023. arXiv:2309.04911 [cs]. URL: <http://arxiv.org/abs/2309.04911>, doi:10.48550/arXiv.2309.04911.
- [BW24] Jeanine Banks and Tris Warkentin. Gemma: Introducing new state-of-the-art open models, Feb 2024. URL: <https://blog.google/technology/developers/gemma-open-models/>.
- [CLR⁺18] Alexis Conneau, Guillaume Lample, Marc’Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word Translation Without Parallel Data, January 2018. arXiv:1710.04087. URL: <http://arxiv.org/abs/1710.04087>, doi:10.48550/arXiv.1710.04087.
- [fac22] facebookresearch. Can faiss be installed with conda on jetson? · issue #2316 · facebookresearch/faiss, 2022. URL: <https://github.com/facebookresearch/faiss/issues/2316>.
- [GGH⁺24] Sukhpal Singh Gill, Muhammed Golec, Jianmin Hu, Minxian Xu, Junhui Du, Huaming Wu, Guneet Kaur Walia, Subramaniam Subramanian Murugesan, Babar Ali, Mohit Kumar, Kejiang Ye, Prabal Verma, Surendra Kumar, Felix Cuadrado, and Steve Uhlig. Edge AI: A Taxonomy, Systematic Review and Future Directions, October 2024. arXiv:2407.04053. URL: <http://arxiv.org/abs/2407.04053>, doi:10.48550/arXiv.2407.04053.
- [GXG⁺24] 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, March

2024. arXiv:2312.10997 [cs]. URL: <http://arxiv.org/abs/2312.10997>, doi:10.48550/arXiv.2312.10997.
- [GZA⁺23] Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. Textbooks are all you need, Jun 2023. URL: <https://arxiv.org/abs/2306.11644>, doi:10.48550/arXiv.2306.11644.
- [hug18] huggingface. Pipelines, 2018. URL: https://github.com/huggingface/transformers/blob/main/docs/source/en/main_classes/pipelines.md.
- [hug22] huggingface. Hugging face hub documentation, 2022. URL: <https://github.com/huggingface/hub-docs/blob/main/docs/hub/index.md>.
- [hug25] huggingface. Hugging face x langchain : A new partner package, Jan 2025. URL: <https://huggingface.co/blog/langchain>.
- [ILC] ILCOR. International Liaison Committee on Resuscitation. URL: <https://ilcor.org/about>.
- [JLF⁺23] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, DeLong Chen, Wenliang Dai, Ho Shu Chan, Andrea Madotto, and Pascale Fung. Survey of Hallucination in Natural Language Generation. *ACM Computing Surveys*, 55(12):1–38, December 2023. arXiv:2202.03629 [cs]. URL: <http://arxiv.org/abs/2202.03629>, doi:10.1145/3571730.
- [Lan23] LangChain. Langchain + chroma, Feb 2023. URL: <https://blog.langchain.dev/langchain-chroma/>.
- [Lan24] LangChain. Introduction — langchain, 2024. URL: <https://python.langchain.com/docs/introduction/>.
- [LBE⁺23] Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. Textbooks are all you need ii: phi-1.5 technical report. *arXiv (Cornell University)*, Sep 2023. doi:10.48550/arxiv.2309.05463.
- [LLC⁺24] Zhenyan Lu, Xiang Li, Dongqi Cai, Rongjie Yi, Fangming Liu, Xiwen Zhang, Nicholas D. Lane, and Mengwei Xu. Small Language Models: Survey, Measurements, and Insights, September 2024. arXiv:2409.15790 [cs]. URL: <http://arxiv.org/abs/2409.15790>, doi:10.48550/arXiv.2409.15790.

- [LPP⁺21] 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, April 2021. arXiv:2005.11401 [cs]. URL: <http://arxiv.org/abs/2005.11401>, doi:10.48550/arXiv.2005.11401.
- [Mak19] Mike Makers. Nvidia Jetson Nano developer kit: A \$99 board Low cost board for Artificial intelligence, April 2019. URL: <https://www.makersgeneration.net/single-post/2019/04/17/nvidia-jetson-nano-developer-kit-a-99-board-low-cost-board-for-artificial-intelligence>.
- [Mau24] Anoop Maurya. The indispensable reranker: A deep dive into optimizing retrieval in retrieval-augmented generation (rag) systems:, May 2024. URL: <https://medium.com/@mauryaanoop3/the-indispensable-reranker-a-deep-dive-into-optimizing-retrieval-in-retrieval-augmented-generation-a12793ee393c>.
- [MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space, September 2013. arXiv:1301.3781. URL: <http://arxiv.org/abs/1301.3781>, doi:10.48550/arXiv.1301.3781.
- [MNB⁺15] Koenraad G. Monsieurs, Jerry P. Nolan, Leo L. Bossaert, Robert Greif, Ian K. Maconochie, Nikolaos I. Nikolaou, Gavin D. Perkins, Jasmeet Soar, Anatolij Truhlář, Jonathan Wyllie, David A. Zideman, Gamal Eldin Abbas Khalifa, Annette Alfonzo, Hans-Richard Arntz, Helen Askitopoulou, Abdelouahab Bellou, Farzin Beygui, Dominique Biarent, Robert Bingham, Joost J. L. M. Bierens, Bernd W. Böttiger, Leo L. Bossaert, Guttorm Brattebø, Hermann Brugger, Jos Bruinenberg, Alain Cariou, Pierre Carli, Pascal Casan, Maaret Castrén, Athanasios F. Chalkias, Patricia Conaghan, Charles D. Deakin, Emmy D. J. De Buck, Joel Dunning, Wiebe De Vries, Thomas R. Evans, Christoph Eich, Jan-Thorsten Gräsner, Robert Greif, Christina M. Hafner, Anthony J. Handley, Kirstie L. Haywood, Silvija Hunyadi-Antičević, Rudolph W. Koster, Anne Lippert, David J. Lockey, Andrew S. Lockey, Jesús López-Herce, Carsten Lott, Ian K. Maconochie, Spyros D. Mentzelopoulos, Daniel Meyran, Koenraad G. Monsieurs, Nikolaos I. Nikolaou, Jerry P. Nolan, Theresa Olasveengen, Peter Paal, Tommaso Pellis, Gavin D. Perkins, Thomas Rajka, Violetta I. Raffay, Giuseppe Ristagno, Antonio Rodríguez-Núñez, Charles Christoph Roehr, Mario Rüdiger, Claudio Sandroni, Susanne Schunder-Tatzber, Eunice M. Singletary, Markus B. Skrifvars, Gary B. Smith,

- Michael A. Smyth, Jasmeet Soar, Karl-Christian Thies, Daniele Trevisanuto, Anatolij Truhlář, Philippe G. Vandekerckhove, Patrick Van de Voorde, Kjetil Sunde, Berndt Urlesberger, Volker Wenzel, Jonathan Wyllie, Theodoros T. Xanthos, and David A. Zideman. European Resuscitation Council Guidelines for Resuscitation 2015: Section 1. Executive summary. *Resuscitation*, 95:1–80, October 2015. URL: <https://www.sciencedirect.com/science/article/pii/S0300957215003500>, doi:10.1016/j.resuscitation.2015.07.038.
- [MSM⁺22] Nils Mühlenfeld, Philipp Störmann, Ingo Marzi, Felix Rosenow, Adam Strzelczyk, René D. Verboket, and Laurent M. Willems. Seizure related injuries – Frequent injury patterns, hospitalization and therapeutic aspects. *Chinese Journal of Traumatology*, 25(5):272–276, September 2022. URL: <https://www.sciencedirect.com/science/article/pii/S1008127521001619>, doi:10.1016/j.cjte.2021.10.003.
- [MSV19] Janakiram MSV. Nvidia launches jetson nano - a competitor to google coral and intel up squared, Mar 2019. URL: <https://www.forbes.com/sites/janakirammsv/2019/03/19/nvidia-launches-jetson-nano-a-competitor-to-google-coral-and-intel-up-squared/>.
- [Mud23] Sujatha Mudadla. What is cross encoder? - sujatha mudadla - medium, Nov 2023. URL: <https://medium.com/@sujathamudadla1213/what-is-cross-encoder-fec22b58f16c>.
- [Nef18] Alarcon Nefi. NVIDIA Jetson AGX Xavier Developer Kit Now Available, September 2018. URL: <https://developer.nvidia.com/blog/nvidia-jetson-agx-xavier-developer-kit-now-available/>.
- [Nid24] Nidhiworah. Chroma db- introduction - nidhiworah - medium, Jun 2024. URL: <https://medium.com/@nidhiworah02/chroma-db-introduction-25718915bae6>.
- [NVIa] NVIDIA. JetPack Archive. URL: <https://developer.nvidia.com/embedded/jetpack-archive>.
- [NVib] NVIDIA. Jetson Nano. URL: <https://developer.nvidia.com/embedded/jetson-nano>.
- [NVI21] NVIDIA. Macchine autonome scalabili basate su ia — nvidia jetson agx xavier, October 2021. URL: <https://web.archive.org/web/20211004094732/https://www.nvidia.com/it-it/autonomous-machines/embedded-systems/jetson-agx-xavier/>.

- [Ope] OpenAI. GPT-2: 1.5B release. URL: <https://openai.com/index/gpt-2-1-5b-release/>.
- [PNI⁺18] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations, March 2018. arXiv:1802.05365. URL: <http://arxiv.org/abs/1802.05365>, doi:10.48550/arXiv.1802.05365.
- [PP19] Usman Pirzada and Usman Pirzada. Nvidia announces the jetson nano – a \$99 low power computer capable of ai loads, Mar 2019. URL: <https://wccftech.com/nvidia-announces-the-jetson-nano-a-99-low-power-computer-capable-of-ai-loads/>.
- [QSX⁺21] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. Pre-trained Models for Natural Language Processing: A Survey, June 2021. arXiv:2003.08271. URL: <http://arxiv.org/abs/2003.08271>, doi:10.48550/arXiv.2003.08271.
- [Res] Restack. Ollama compatibility with cuda 12.5 — restackio. URL: <https://www.restack.io/p/ollama-answer-cuda-12-5-compatibility-cat-ai#cm1trpwr070h26hu0mwxrf8y>.
- [SGM19] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and Policy Considerations for Deep Learning in NLP, June 2019. arXiv:1906.02243 [cs]. URL: <http://arxiv.org/abs/1906.02243>, doi:10.48550/arXiv.1906.02243.
- [Sim24] Nisanth Simon. Faiss (facebook ai similarity search) : A deep dive into index structures for efficient data exploration, Aug 2024. URL: <https://medium.com/@nisanth.m.s/faiss-facebook-ai-similarity-search-a-deep-dive-into-index-structures-for-efficient-data-7a4524037556>.
- [Sis23] Yogendra Sisodia. State-of-the-art bge embeddings for retrieval augmented generation, Aug 2023. URL: <https://medium.com/@scholarly360/state-of-the-art-bge-embeddings-for-retrieval-augmented-generation-2c882afb9e37>.
- [SLBK24] Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A Simple and Effective Pruning Approach for Large Language Models, May 2024. arXiv:2306.11695. URL: <http://arxiv.org/abs/2306.11695>, doi:10.48550/arXiv.2306.11695.

- [SPC⁺21] Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. Retrieval Augmentation Reduces Hallucination in Conversation, April 2021. arXiv:2104.07567 [cs]. URL: <http://arxiv.org/abs/2104.07567>, doi:10.48550/arXiv.2104.07567.
- [Str24] Marva Stroganova. How to Read and Make Box Plot: A Complete Guide + Best Practices, March 2024. URL: <https://mlsamurai.medium.com/how-to-read-and-make-box-plot-a-complete-guide-best-practices-92b233e59c3b>.
- [suh21] suhash. Jetson Software Roadmap for 2H-2021 and 2022, May 2021. URL: <https://forums.developer.nvidia.com/t/jetson-software-roadmap-for-2h-2021-and-2022/177721>.
- [Tea24a] Qwen Team. Hello qwen2, Jun 2024. URL: <https://qwenlm.github.io/blog/qwen2/>.
- [Tea24b] Qwen Team. Introducing qwen1.5, Feb 2024. URL: <https://qwenlm.github.io/blog/qwen1.5/>.
- [Tea24c] Qwen Team. Qwen2.5: A party of foundation models!, Sep 2024. URL: <https://qwenlm.github.io/blog/qwen2.5/>.
- [Tea24d] Qwen Team. Qwen/Qwen2.5-0.5B-Instruct · Hugging Face, November 2024. URL: <https://huggingface.co/Qwen/Qwen2.5-0.5B-Instruct>.
- [Uch19] Kristin Uchiyama. Nvidia announces jetson nano: \$99 tiny, yet mighty nvidia cuda-x ai computer that runs all ai models, Mar 2019. URL: <http://nvidianews.nvidia.com/news/nvidia-announces-jetson-nano-99-tiny-yet-mighty-nvidia-cuda-x-ai-computer-that-runs-all-ai-models>.
- [VSP⁺23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, August 2023. arXiv:1706.03762. URL: <http://arxiv.org/abs/1706.03762>, doi:10.48550/arXiv.1706.03762.
- [WCL⁺24] Jingcun Wang, Yu-Guang Chen, Ing-Chao Lin, Bing Li, and Grace Li Zhang. Basis sharing: Cross-layer parameter sharing for large language model compression, October 2024. arXiv:2410.03765. URL: <http://arxiv.org/abs/2410.03765>, doi:10.48550/arXiv.2410.03765.
- [Wik24] Wikipedia. Good Samaritan law, December 2024. Page Version ID: 1263637414. URL: https://en.wikipedia.org/w/index.php?title=Good_Samaritan_law&oldid=1263637414.

- [Wik25] Wikipedia. Soccorritore, January 2025. Page Version ID: 143119212. URL: <https://it.wikipedia.org/w/index.php?title=Soccorritore&oldid=143119212>.
- [YGZ⁺24] Hao Yu, Aoran Gan, Kai Zhang, Shiwei Tong, Qi Liu, and Zhaofeng Liu. Evaluation of Retrieval-Augmented Generation: A Survey, July 2024. arXiv:2405.07437 version: 2. URL: <http://arxiv.org/abs/2405.07437>, doi:10.48550/arXiv.2405.07437.