FernUniversität in Hagen
Faculty of Mathematics and Computer Science

Artificial Intelligence Group

Analysis of Offline Large Language Models as AI Help Desk Using Real Customer Data

Bachelor's Thesis

in partial fulfillment of the requirements for
the degree of Bachelor of Science (B.Sc.)
in Informatik

submitted by
Malte Siems

First examiner: Prof. Dr. Matthias Thimm
Artificial Intelligence Group

Advisor: Dr. Maik Wischow

Statement

I declare that I have written the bachelor´s thesis independently and without unauthorized use of third parties. I have only used the indicated resources and I have clearly marked the passages taken verbatim or in the sense of these resources as such. The assurance of independent work also applies to any drawings, sketches or graphical representations. The work has not previously been submitted in the same or similar form to the same or another examination authority and has not been published. By submitting the electronic version of the final version of the bachelor´s thesis, I acknowledge that it will be checked by a plagiarism detection service to check for plagiarism and that it will be stored exclusively for examination purposes.

I explicitly agree to have this thesis published on the webpage of the artificial intelligence group and endorse its public availability.

The software and associated research data for this work are available at the AIG for viewing upon request.

Potsdam, 05.10.25 M. Siems

(Place, Date) (Signature)

Zusammenfassung

Konventionell organisierter Kundenservice ist effektiv, aber personal- und somit kostenintensiv. Chatbots auf der Basis von Large Language Models (LLMs) bieten großes Potential, den Kundensupport von Unternehmen zu verbessern, wenn die zentralen Herausforderungen der Antwortqualität und Datensicherheit adäquat adressiert werden.

In der vorliegenden Abschlussarbeit wurden verschiedene Implementierungsansätze von Chatbots auf Basis von *Retrieval Augmented Generation (RAG)* untersucht. In einer Machbarkeitsstudie wurden verschiedene *Retrieval*-Strategien und Modelle für die Generierung verglichen. Als Fallstudie dafür diente die Supportabteilung einer Firma für Softwareprodukte zur Entwicklung mobiler Arbeitsmaschinen. Als Datenbasis für RAG dienten Handbücher und Softwarebeschreibungen, die Text und Abbildungen enthielten. Zur Verarbeitung beider Modalitäten kamen *Vision-Language-Models (VLMs)* zum Einsatz. Aus Datenschutzgründen wurde das System ausschließlich lokal betrieben.

Auf der Basis von multimodalen Benchmarks wurden VLMs ausgewählt und reale Kundenanfragen wurden anonymisiert und als Testdaten verwendet. Domänenexperten bewerteten die Antworten anhand der Kriterien Nützlichkeit, Korrektheit, Klarheit und Prägnanz. Eine darauffolgende statistische und qualitative Analyse führte zu folgenden Ergebnissen:

Die unterschiedlichen *Retrieval*-Strategien durch *dense* und *sparse* Methoden ergänzen sich gegenseitig, da sie in jeweils unterschiedlichen Fällen scheitern. Als zentral hat sich dabei die Methode zur Priorisierung der gefundenen Passagen herauskristallisiert, da die hier angewandte gewichtete Summe aus normalisierten Ähnlichkeitswerten teilweise zu unvorteilhaften Einstufungen führte. Die in dieser Studie untersuchten Modelle zur Generierung der Antworten zeigten ähnliche Leistung. Lediglich die Quantisierung eines Modells mit höherer Parameteranzahl führte zu besseren Bewertungen verglichen mit dem Gegenstück gleicher Größe. Unter allen Modellen war zu beobachten, dass diese die gegebenen Informationen falsch einschätzten, was in der Verknüpfung von nicht in Beziehung stehenden Konzepten oder dem Ignorieren von essentiellen Aspekten deutlich wurde. In 15 von 65 Fällen (23%) konnte eine hilfreiche Antwort generiert werden, was zeigt, dass das System verbessert werden muss, bevor es praktische Anwendung finden kann. Zukünftige Untersuchungen sollten die möglichen Potenziale wie die Reformulierung der Anfragen, der Einsatz von *Cross-Encodern* zur Bewertung der gefundenen Informationen und der Einsatz komplexer RAG Methoden, z.B. auf Basis von Rekursion, untersuchen.

Abstract

Conventional customer support is effective, but labor-intensive and therefore costly. Chatbots based on Large Language Models (LLMs) have great potential to enhance companies' customer service. Key challenges in their use include the quality of answers and concerns about data protection when processing sensitive information.

Within this thesis, different chatbot implementations utilizing Retrieval Augmented Generation (RAG) were examined. A feasibility study was conducted to compare multiple retrieval approaches and generator models within the customer support domain of a company selling software for mobile machinery development. The database for retrieval comprised software documentation and manuals containing text and images, which required the use of Vision-Language-Models (VLMs). For data protection reasons, the whole pipeline was operated offline.

VLMs were selected by analyzing multimodal benchmarks. Actual customer requests were anonymized and used as test cases to evaluate the response quality. Domain experts assessed the answers based on the predefined criteria: helpfulness, correctness, clarity, and conciseness. Subsequently, a statistical and qualitative analysis was conducted revealing these findings:

The variation of retrieval strategies showed that dense and sparse methods complement each other, as they fail in different cases. It was observed that the fusion strategy is crucial to rank the chunks retrieved with both methods, as the applied approach to normalize and combine similarity scores with a weighted sum failed occasionally. The examined models used for generation performed equally, while a quantized model with higher parameter count outperformed its counterpart of the same size. However, all generators misjudged the given context by either ignoring important aspects or linking unrelated concepts. In 15 out of 65 cases (23%), a helpful answer could be provided by the RAG setup, indicating that it needs enhancement before practical deployment. Future research should examine potential improvements such as query rewriting, more sophisticated fusion strategies through cross-encoding, and advanced RAG methods like recursive retrieval.

Contents

List of Figures

List of Tables

Glossary

Attention Layer	A core component of the transformer architecture that allows to weigh the importance of different parts of the input data for each output element, *see* Section 2.3.2
BM25	Best Match 25: An algorithm to assess similarity of texts based on lexical matching, *see* Section 2.2.2
Chunking Strategy	The method used to divide large documents into smaller, manageable, and semantically coherent sections (chunks) for efficient retrieval (in this thesis: hierarchical chunking considering document headings)
Cross-Encoder	A component to refine the relevance ranking of documents retrieved from a database by processing the query and the document text side-by-side
Dense Embedding	A representation technique of data, where text and images are represented by low-dimensional vectors capturing semantics, *see* Section 2.2.2
Dense Retrieval	A retrieval strategy which utilizes dense embeddings, *see also* Dense Embedding, Section 2.5.2
Embedding	The numerical vector representations of data used in language processing, *see also* Section 2.2.2
Fine-Tuning	Refers to Parameter-Efficient Fine-tuning, the process of adapting a pre-trained model to a specific task or dataset, *see* Section 2.5.1
Friedman two-way analysis of variance	A non-parametric test used to detect statistically significant differences across multiple related datasets, *see* Section 3.5
Fusion Strategy	The method used in hybrid retrieval to combine similarity scores from different retrieval methods (e.g., dense and sparse) to rerank chunks, *see also* Hybrid Retrieval

Generator	A major component in the RAG pipeline (in this thesis: a VLM) that formulates the final answer considering the query and provided context, *see also* VLM
Holm-Bonferroni Correction	A technique to control the family-wise error rate, preventing an increase in Type I errors when multiple statistical tests are conducted on the same samples, *see* Section 3.6
Hybrid Retrieval	A strategy combining both dense and sparse retrieval methods to leverage their respective strengths in finding relevant documents, *see also* Dense Retrieval, Sparse Retrieval, Section 2.5.2
Krippendorff's Alpha	A statistical measure to assess annotator agreement, *see* Section 3.7
KV-Cache	A memory structure in transformer models that stores the calculated Key and Value vectors of previous tokens to speed up the generation of the next token, *see also* Attention Layer
LLM	Large Language Model: A deep learning model trained to understand and generate human language, *see* Section 2.4
Multimodal	Pertaining to systems or data that involve multiple modes of input/output (in this thesis: text and image)
Ordinal Data	Data that has an intrinsic ordering or sequence (e.g., ratings like helpfulness 1-5) but where the distance between values is not strictly equal, *see* Section 3.1
Parameter	In this thesis, the term is used as a synonym for the adjustable weight in an artificial neuron (e.g., in LLMs and VLMs), which determines the strength of the connection between inputs and the neuron's output
Quantization	The process of reducing the numerical precision (e.g., from 32-bit to 4-bit) of a model's weights

Query Rewriting	A technique for potential RAG improvement where the initial user query is refined or decomposed into simpler subqueries to improve retrieval performance
RAG	Retrieval Augmented Generation: Before generation with a language model, context retrieved from a knowledge database is dynamically added to the prompt to produce more grounded answers
Retriever	A major component in the RAG pipeline that searches the database for chunks of information relevant to the user's query, *see also* Sparse Retrieval, Dense Retrieval, Hybrid Retrieval
Sparse Embedding	A vectorization technique, typically with high dimensionality, where most entries in the vector are zero (e.g., TF-IDF), *see* Section 2.2.2
Sparse Retrieval	A retrieval strategy which utilizes sparse embeddings, *see also* Sparse Embedding, Section 2.5.2
Token	The smallest, atomic unit of text (often a word, subword, or character) used in language processing, *see* 2.2.1
VLM	Vision Language Model: In this thesis, the term is used for models which can comprehend text and images and generate text (text-image-to-text)
Wilcoxon matched-pairs signed-ranks test	A non-parametric test used to examine two dependent samples for statistically significant differences, which considers the magnitude of the differences, *see* Section 3.4

1. Introduction

Serving as the reader's orientation, this chapter motivates the thesis by establishing the context and identifying the research gap. Subsequently, it articulates the research questions and provides an overview of the approach used to address them, which thereby sets the document's structure.

1.1. Motivation

Customer service should adhere to the old mantra 'the customer is king.' Loosely interpreted, this means customer needs should be satisfied unconditionally. In the context of software products, customer service plays a crucial role, especially when it comes to solving problems while the software is in use. However, US companies lose approximately \$1.6 trillion annually due to the consequences of inadequate support [SJA23]. A study indicates that customers who have a positive service experience remain loyal to the company in 90% of cases and recommend it to others [SJA23]. Investment in good customer service therefore pays off, but if organized traditionally, it is very labor-intensive and therefore costly.

With the rapid development of Large Language Models (LLM), there is significant potential to make customer support more cost-efficient [Chk+24]. Semi-automated communication with customers via chatbots increases availability and allows employees to focus on more complex tasks [Ina24]. The challenges here are the need for a high-quality and up-to-date database, and compliance with data protection [Rus+22].

The basic capabilities of LLMs such as text comprehension and mathematics are evaluated using standard tests such as Massive Multitask Language Understanding [Hen+20] or Grade School Math 8K [Cob+21]. While numerous benchmarks exist to evaluate the fundamentals, publicly available performance evaluation on real-world tasks is limited. To bridge this gap, this bachelor's thesis is intended to make a research contribution with a feasibility study to evaluate the performance of models in a real-world application context. For this purpose, various concepts of LLM chatbots were developed and compared with each other.

1.2. Research Focus

The support division of a company[1] that sells software tools assisting in the development of mobile machinery served as a case study. Due to data protection, the study is limited to offline LLMs. The knowledge base consists of manuals and technical documentation. This data contains images, tables, and screenshots of software, making LLMs with image processing capabilities – so-called Vision Language Models (VLMs) [Bor+24] – necessary.

[1]The company name has been anonymized.

To make the domain-specific data accessible to the models, two state-of-the-art approaches can be used: (Parameter-Efficient) Fine-Tuning and Retrieval Augmented Generation (RAG) [SKH24]. The former technique requires huge amounts of training data to adjust either the models or additionally introduced parameters (weights). This training data was not available in the described use case and therefore the latter approach, RAG, was applied. It is based on augmenting the initial prompt with retrieved information from a database relevant to the prompt's context. As a result, the model generates a more sophisticated answer by processing the additional data.

In detail, the proposed chatbot design can be varied regarding different aspects:

- the VLM itself,
- the VLMs size, i.e. the number of weights (some models come in different sizes),
- the precision of each weight (quantization),
- the exact RAG implementation (e.g. dense vs. sparse embedding, see Section 2.2.2).

It is important to note that these core aspects are just a selection of possible design choices, which are subject of this work. Other variations could concern the data chunking strategy, prompting techniques (prompt engineering), and query transformation.

This feasibility study is structured as follows: Different chatbot designs are developed to answer actual, anonymized customer inquiries. The answers are then evaluated by domain experts on the basis of previously developed criteria. Finally, the results are statistically and qualitatively analyzed to answer the following research questions:

- What is the response quality of three selected, state-of-the-art VLMs when solving customer inquiries about a software product for mobile machinery development?
- Using one model as a case study, how does varying the model's number of parameters (weights) and weight quantization affect the response quality?
- How do different knowledge embedding approaches affect the response quality (dense embedding, sparse embedding or hybrid approach)?

1.3. Outline

This thesis is structured as follows: Section 1 introduces the research problem and outlines the overall approach. The technical fundamentals of language processing and RAG are established in Section 2, while Section 3 introduces statistical methods for appropriate analysis. Together, they cover the required background knowledge and therefore build the foundation for this thesis. Related research is reviewed in

Section 4 and benchmarks used for the preselection of VLMs are introduced. Subsequently, the use of human evaluation is motivated, and Section 5 provides implementation details of the RAG pipeline and the annotation process conducted. The results are presented and statistically analyzed in Section 6. The discussion and quantitative analysis in Section 7 reveal the core findings and contextualize them. Furthermore, the study's limitations are critically examined, and the section concludes with an overview of critical points of failure. Finally, the findings are summed up, and recommendations for future research are given in Section 8.

2. Fundamentals of Language Processing

This section presents the technical fundamentals of this thesis by introducing core concepts of language processing. It starts from basics of neural networks (Section 2.1), through core concepts such as tokens and vector representations of language (Section 2.2), toward language processing with feed-forward and attention mechanisms (Section 2.3). The main architectures – encoder-decoder, encoder, and decoder – are presented (Section 2.4) and complemented with advanced methods like Parameter-Efficient Fine-Tuning, Retrieval Augmented Generation, and Prompt Engineering (Section 2.5).

2.1. Neural Networks

The following subsections are divided into the introduction of biological and artificial neurons, the expansion to neural networks, and finally a brief on training. They are based on the work of Gurney [Gur97].

2.1.1. Biological and Artificial Neurons

The structure and operating principle of artificial neurons are inspired by their biological counterparts. A biological neuron is a type of cell specialized in sending electrical impulses as part of the nervous system [Ram99]. In simplified terms, it consists of the cell body from which dendrites and the axon extend (see Figure 1). The dendrites are connected to neighboring cells and receive electrical impulses, which are aggregated at the axon hillock, the area where the cell body and the axon merge. The electrical impulses induce a potential at the axon hillock. When this potential exceeds a certain threshold, an electrical impulse travels down the axon to reach other neurons or muscles.

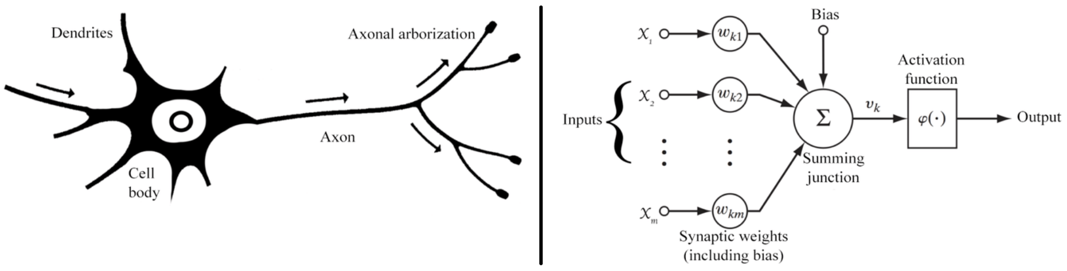


Figure 1: Structure of biological neuron (left) and artificial neuron (right) [AD18]

An artificial neuron consists of two components, often referred to as pre-activation and activation function [MP43]. In analogy to the biological neuron, the pre-activation aggregates inputs x_i via a weighted sum, while an additional bias b is introduced:

$$v(x) = \sum_{i=0}^{m} x_i w_i + b = \boldsymbol{x}^\top \boldsymbol{w} + b. \tag{1}$$

The activation function implements a threshold and thus introduces nonlinearity (more details on this in the Section 2.1.2). An easy implementation of the activation function is the Rectified Linear Unit (ReLU), defined as:

$$ReLU(x) = max(0, x), \tag{2}$$

which only activates if the output is positive.

Interpreting the inputs $\boldsymbol{x}$ and the weights $\boldsymbol{w}$ as vectors, the dot product between them is a measure for similarity. In geometrical space, the dot product is positive if the angle between two vectors is less than 90°, negative if the angle exceeds 90°, and is zero at 90°. Therefore, the neuron is only activated, if the input and weight vectors are aligned to some extent. Equation 1 can also be interpreted as a hyperplane separating into an activation and non-activation space.

2.1.2. Multi-Layer Network Architectures

To model complex reasoning, numerous neurons are connected in neural networks, which are structured in layers of parallel neurons [MP43]. Each layer uses the previous layer's outputs as inputs, and the weight vector of Equation 1 expands to a matrix, while the bias scalar becomes a vector accordingly. The number of layers refers to the network's depth and the quantity of parallel neurons per layer to its width. At this point, the importance of nonlinearity of the activation function must be emphasized. Without the nonlinearity, stacked layers would collapse to one single layer, because concatenated linear operations can be expressed as one single operation.

The optimal design of a network regarding depth and width is subject to current research (see [NRK21], [SS17]) and will not be further discussed here. Instead, a more descriptive approach is presented. If a single neuron can be interpreted as a hyperplane (see Section 2.1.1), then a parallel set of neurons can form a more complex boundary to decide on activation. Consider a scenario where a neural network is trained and used for recognizing geometrical shapes of different colors in images. Simplified and idealized, neurons of the first layer could identify dominant color ranges and features like edges and curves. In a second layer, neurons get activated when multiple of these features of the first layer are activated to maybe detect parallel lines of specific color or angles between lines. In a third layer, this information is combined to the final classification of, for example, a red square.

2.1.3. Training

A neuron's activation function fires if the dot product of weight and input vector exceeds a certain threshold. To tailor a neuron to a specific behavior, it must be trained on datasets, each consisting of an input vector $\boldsymbol{x}$ and scalar target t. The weights $\boldsymbol{w}$ can be adjusted via the equation:

$$\boldsymbol{w}_{new} = \boldsymbol{w}_{old} + \alpha(t - y)\boldsymbol{x}, \tag{3}$$

with scalar y being the neuron's output and $\alpha \in (0, 1)$ a training parameter. If the target and actual output differ, the weights are adjusted, otherwise, they stay constant.

In neural networks, the inputs and targets of middle layer neurons are not explicitly known, making the use of Equation 3 impossible. Therefore, more advanced approaches were developed. One widely-used training method is referred to as backpropagation, which is briefly explained below. In backpropagation, the network's scalar error E is usually defined as half of the sum of squared errors (SSE):

$$E = \frac{1}{2}\sum_{i=1}^{m}(t_i - y_i)^2, \tag{4}$$

where factor $\frac{1}{2}$ is used for convenience when building the derivation later, and m is the width of the output layer. Each scalar weight w is adjusted according to the gradient descent method:

$$w_{new} = w_{old} - \alpha\frac{\partial E}{\partial w}, \tag{5}$$

where, analogous to Equation 3, α is a learning parameter. To ascertain each individual weight's influence on the overall error, the chain rule is used to calculate the gradient:

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial v_j}\frac{\partial v_j}{\partial w_i}, \tag{6}$$

for a weight i of a node j. Each fraction can then be determined separately (use of ReLU as activation function is assumed):

$$\frac{\partial E}{\partial y_j} = y_j - t_j \tag{7}$$

$$\frac{\partial y_j}{\partial v_j} = \begin{cases} 0 & \text{if } v_j < 0 \\ 1 & \text{else} \end{cases} \tag{8}$$

$$\frac{\partial v_j}{\partial w_i} = \frac{\partial(x_0 w_0 + x_1 w_1 + \cdots + x_m w_m + b)}{\partial w_i} = x_i. \tag{9}$$

After all gradients of the output layer are calculated, the gradients of the second-last layer and so on can be computed in a recursive manner. The backpropagation therefore works in the opposite direction of the network's activation. In this way, all weights can be adjusted regarding their contribution to the total error.

2.2. Core Concepts of Language Modeling

Neural networks are the fundamentals of language processing. Before introducing latter in the following subsection, some core concepts of language modeling are presented here.

2.2.1. Token and Tokenization

In language modeling, text is usually not processed as a whole, but is rather divided into smaller pieces first. The smallest, atomic unit of text is called a token.

Tokens can be words, partial words or even smaller units down to individual letters and punctuation marks [Nav+24]. Tokenization is the process by which text is broken down into tokens. There are a number of techniques for this. One of the simplest methods divides the text into words at each space, for example the phrase 'I write a thesis' would be broken down into the four tokens 'I', 'write', 'a', 'thesis'. Nowadays, more complex algorithms are state of the art, such as byte pair encoding and word piece encoding, which lead to a finer granularity [Min+25]. These techniques break down complex words like 'unbelievable' into 'un', 'bel', 'ieve', 'able'[2].

2.2.2. Vectorization

Vectorization is a process often taking place after the tokenization of a text. Each token is assigned a vector representation, referred to as an embedding. Depending on the exact task in language processing, there are different approaches to determine these vectors in the first place. Some are briefly presented in the next paragraphs.

Bag of Words The following paragraph is based on the work of Juluru et al. [Jul+21]. Bag of words (BoW) is based on the idea that a text can be characterized by often occurring words. Therefore, each word or token of the vocabulary is assigned one dimension in vector space. A text can then be represented by a vector with entries corresponding to the word count while information on word order is lost.

Usually, a text is pre-processed in the first place to normalize all words to their dictionary form, and stop words like 'the' or 'is' are removed.

Term Frequency (TF) and Inverse Document Frequency (IDF) Instead of counting absolute occurrence of terms, the concept is extended to using the term frequency $TF_{t,d}$, i.e., the fraction of count of a term t and total number of terms inside a document d:

$$TF_{t,d} = \frac{\text{\# term } t \text{ in document } d}{\text{\# total terms in document } d}. \tag{10}$$

[2]The tokenizer playground `https://huggingface.co/spaces/Xenova/the-tokenizer-playground` allows exploring tokenization used in selected LLMs. For the example, the tokenizer of GPT-4 was used.

To take into account that some terms might be common along specific context but are not suitable for distinction, inverse document frequency IDF_t has been developed. It scores the frequency of how often a term occurs over several documents:

$$IDF_t = \log \frac{\text{\# documents}}{\text{\# documents containing term } t}. \tag{11}$$

As a result, if a term occurs in all documents, the IDF-score is zero. TF-IDF combines both approaches through multiplication. Therefore, a term is represented in the vector, if it is of high frequency within one document and at the same time unique along all documents [MRS08].

Best Match 25 (BM25) In the context of similarity search, TF-IDF introduces a bias towards long documents, because higher term frequency is more likely in longer texts. The Best Match 25 (BM25) addresses this issue through document length normalization. Additionally, it introduces some major refinements on Term Frequency calculation. It is based on the assumption that the relevance of the Term Frequency saturates with increasing occurrence. A concise representation of the scalar BM25 score for a term t in document d is:

$$BM25_{t,d} = \frac{TF_{t,d}}{k_1((1-b) + b\frac{L_d}{L_{avg}}) + TF_{t,d}} w_i^{RSJ}, \tag{12}$$

with document length L_d, average document length L_{avg}, and the scalar Robertson-Spärck Jones weight w_i^{RSJ}, which is a refined version of IDF. The constant k_1 determines how fast TF saturates, while $0 \leq b \leq 1$ defines how severely the document length is taken into account.

A final similarity score S between a query q and a document d can then be computed through summation over all terms t, which are in both the query and the document:

$$S_{q,d} = \sum_{t \in q \cap d} BM25_{t,d}. \tag{13}$$

BoW, TF-IDF, and BM25 are simple approaches relying on frequency of words. The determined high-dimensional vector representations are usually sparse, because most texts do not contain nearly all words in a vocabulary. Both approaches work well for retrieval of information, clustering or similarity comparison of documents [RZ09].

Static Word Embeddings Instead of relying on frequency of words, Mikolov et al. proposed Word2Vec using a neural network to extract vector embeddings of tokens through training. The model is trained to predict context words based on a target word and vice versa, one word must be predicted based on context words in a sliding window. The idea behind this is to capture semantic and syntactic relationships of tokens in dense vectors. The relation between well trained representations can be verified by vector sums: The embeddings of Paris minus France, plus Italy, should equal Rome [Mik+13].

Transformer Based Embeddings Static word embeddings cannot distinguish between different scenarios, for example changes in meaning of 'bank' in context of 'river' or 'finance'. State-of-the-art language processing uses mechanisms like attention (see Section 2.3.2) to take into account context, to compute embeddings dynamically. For more on that see the following section on transformer architecture (Section 2.4).

2.2.3. Positional Encoding (PE)

By representing the tokens using vector embeddings, the information about the position or proximity of the individual tokens to each other is lost. However, this information is of great importance in many languages. The idea of PE is to assign a unique embedding to each token in a text, which is only dependent on the position and not the token itself.

Token	Position	Dimension	Formula	Calculation	Value
I	0	0	$\sin(0/10000^{0/4})$	$\sin(0/1) = \sin(0)$	0.000
		1	$\cos(0/10000^{0/4})$	$\cos(0/1) = \cos(0)$	1.000
		2	$\sin(0/10000^{2/4})$	$\sin(0/100) = \sin(0)$	0.000
		3	$\cos(0/10000^{2/4})$	$\cos(0/100) = \cos(0)$	1.000
write	1	0	$\sin(1/10000^{0/4})$	$\sin(1/1) = \sin(1)$	0.841
		1	$\cos(1/10000^{0/4})$	$\cos(1/1) = \cos(1)$	0.540
		2	$\sin(1/10000^{2/4})$	$\sin(1/100) = \sin(0.01)$	0.010
		3	$\cos(1/10000^{2/4})$	$\cos(1/100) = \cos(0.01)$	1.000
a	2	0	$\sin(2/10000^{0/4})$	$\sin(2/1) = \sin(2)$	0.909
		1	$\cos(2/10000^{0/4})$	$\cos(2/1) = \cos(2)$	-0.416
		2	$\sin(2/10000^{2/4})$	$\sin(2/100) = \sin(0.02)$	0.020
		3	$\cos(2/10000^{2/4})$	$\cos(2/100) = \cos(0.02)$	1.000
thesis	3	0	$\sin(3/10000^{0/4})$	$\sin(3/1) = \sin(3)$	0.141
		1	$\cos(3/10000^{0/4})$	$\cos(3/1) = \cos(3)$	-0.990
		2	$\sin(3/10000^{2/4})$	$\sin(3/100) = \sin(0.03)$	0.030
		3	$\cos(3/10000^{2/4})$	$\cos(3/100) = \cos(0.03)$	0.999

Table 2: Example of Sinusoidal Positional Encoding calculations for 'I write a thesis' with $d_{model} = 4$

Various approaches such as Absolute Positional Embedding (APE), Relative Positional Embedding (RPE), and Rotary Positional Embedding (RoPE) can be found in the literature. Here, the Sinusoidal Positional Encoding (SPE) is presented, which is used in the transformer introduced later. The basic idea of SPE is to assign different frequencies to each model dimension and use the position in the sine and cosine function to calculate the final encoding:

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d}}\right) \tag{14}$$

$$PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d}}\right) \tag{15}$$

where pos is the position of a token in the sentence and $i \in \{0, 1, \ldots, d_{\text{model}}/2 - 1\}$ is the index of the embedding's dimension [Vas+17].

Table 2 shows the positional encoding for the example sentence 'I write a thesis', where for simplicity a model dimension of four is assumed and each word is represented by a single token. The final PEs can be gathered from the right column, e.g., for 'write' the embedding would be $(0.841, 0.540, 0.010, 1.000)$.

2.3. Language Processing

Analogous to neural networks, the vector representations of tokens can be processed in layers of neurons. For language processing, the original transformer uses two different types of layers, which are presented in the next subsections.

2.3.1. Feed-Forward

In the feed-forward layer, the input embedding is projected onto a higher dimension, an activation function is applied, and then the dimension is reduced again. This enables the processing of complex relationships [Ras25]. A mathematical formulation with ReLU (see Equation 2) as the activation function is

$$FFN(\boldsymbol{X}) = ReLU(\boldsymbol{X}\boldsymbol{W}_1^\top + \boldsymbol{B}_1)\mathbf{W}_2 + \boldsymbol{B}_2, \tag{16}$$

with weight matrices $\boldsymbol{W}_1$, $\boldsymbol{B}_1$, $\boldsymbol{W}_2$, and $\mathbf{B}_2$ as well as input embedding $\boldsymbol{X}$ [Vas+17].

2.3.2. Attention Layer

The semantics of a word can depend heavily on its context. For example, our association with the word 'bank' changes dramatically if paired with river or finances. The relationship between different words in a sentence also plays an important role in understanding the text, for example, which verb belongs to which subject. To provide these crucial information in language processing, the attention mechanism was developed. In this mechanism, each token is enriched with additional information extracted from surrounding tokens by adjusting the regarded embedding accordingly [Jai22].

Figure 2 shows the flow chart of the attention layer. The matrices $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ serve as inputs, which represent query, key and value. In the specific case of Self-Attention, these matrices are determined by the input embedding of this layer multiplied by three learned weight matrices $\boldsymbol{W}_Q$, $\boldsymbol{W}_K$, and $\boldsymbol{W}_V$:

$$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}_Q \tag{17}$$
$$\boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}_K \tag{18}$$
$$\boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}_V \tag{19}$$

[Ras25].

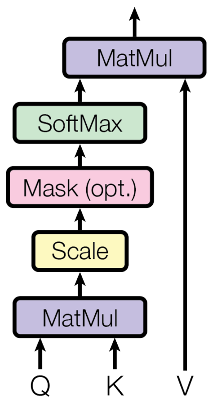


Figure 2: Flow chart of the Attention Mechanism [Vas+17]

Using the example sentence 'I see a black dog', and under the simplification that each word is represented by a single token, the mechanics of these three matrices are explained as follows. In a simplified way, the query of each token basically asks for context, for example, the noun 'dog' might look for adjectives, while the verb 'see' may ask for a subject and object. The key of each token provides answers to these queries, for example, 'black' may signalize that it is an adjective. To test the match between a query and key vector, the dot product is used. For each token, all tokens are examined simultaneously and the matching is calculated. Subsequently, the Softmax function is applied, which is defined as

$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}, \tag{20}$$

providing a scaled distribution of sum one. Looking at the example, the softmax distribution for 'dog' might look like $(0.05, 0.07, 0.01, 0.27, 0.6)$, where each entry corresponds to the token of the sentence. Therefore the words 'dog' and 'black' seem to be important.

The value matrix captures contextually relevant properties of the tokens. For example, when 'black' modifies 'dog', the color aspect might be relevant, whereas when 'black' would modify 'judo belt', the emphasis might lie on the rank or status property.

Finally, a weighted sum of the column vectors of the value matrix is built with the distribution of the softmax function as weights. The result is a new vector representation of 'dog' which now roughly contains the information 'black dog'.

A compact formulation of the attention mechanism is given by:

$$Attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}})\boldsymbol{V}, \tag{21}$$

where d_k is the dimension of the vector embeddings [Vas+17]. The additional scaling reduces the absolute differences of the dot products of queries and keys. The reason for this is the softmax's property to concentrate the attention heavily to the input of highest magnitude, if the absolute differences are substantial.

At this point, two specializations of attention are introduced in the following paragraphs.

Multi-Head Attention In Multi-head Attention, the attention mechanism is executed in parallel for each head h, instead of only once. For this purpose, different weight matrices are provided in the model, which focus on different aspects.

The result of the different heads is concatenated and reduced to the original dimension by an additional matrix multiplication with a learned matrix [Vas+17].

Masked-attention In the attention mechanism, each token is associated with its surrounding context. The process is bidirectional, i.e. both tokens on the left and on the right relative to the token under consideration are taken into account.

If the current token is only to be associated with the left-sided tokens, masking is used, which mathematically represents an addition with a masking matrix M. The entries of this matrix are either zero or minus infinity. The high negative entries lead to probabilities of zero in the downstream softmax function, which means that the entries are effectively masked [Vas+17].

2.4. LLM Architectures

On the basis of the introduced fundamentals of neural networks and language modeling, three standard LLM architectures are presented: Encoder-Decoder, Encoder and Decoder. The basic structure of these three architectures is explained below.

2.4.1. Encoder-Decoder

The Encoder-Decoder architecture (see Figure 3) was first introduced in "Attention Is All You Need" [Vas+17] and is presented below.

The input is first tokenized (see Section 2.2.1) and converted into vector embeddings (see Section 2.2.2). For this, a basic mapping process between the tokens and previously learned vectors is used. Subsequently, the positional encoding (see Section 2.2.3) is calculated and added. The resulting sum serves as the input of the encoder, which is shown in the gray box on the left. It consists of N identical blocks, which are connected sequentially. Each of these blocks consists of a multi-head attention layer (see Section 2.3.2) and a feed-forward layer (see Section 2.3.1). After each layer, its input is added to the output (residual connection) and is then normalized.

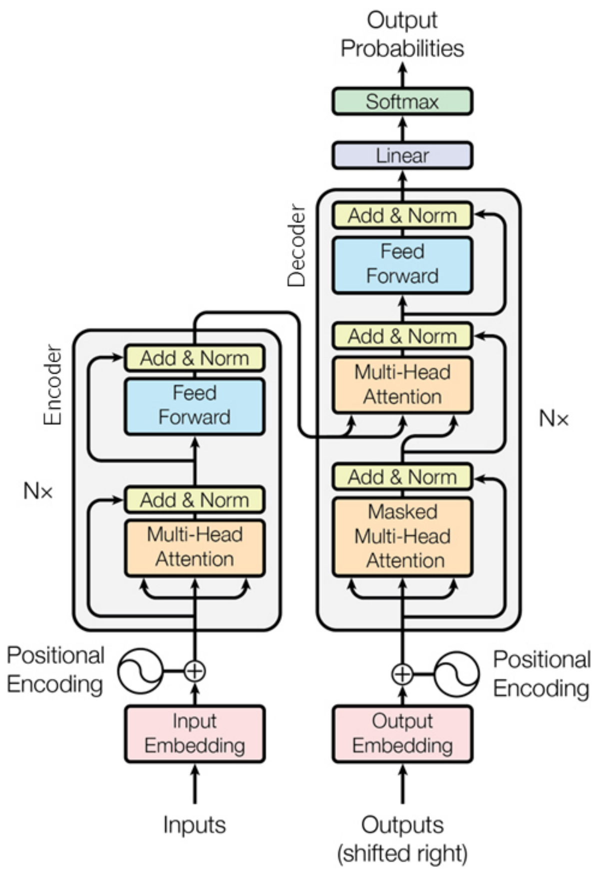


Figure 3: Architecture of the encoder-decoder transformer [Vas+17]

The encoder's attention layer modifies the initial embedding by taking into account the whole context of the sequence. Additionally, complex relationships are introduced by the non-linear feed-forward layer.

The decoder is shown in the right half of Figure 3. Its input is the initially empty output embedding. As with the encoder, the positional encoding is also determined and added in a first step. The structure of the decoder itself is similar to that of the encoder, as it is also structured in N identical blocks, which contain attention and feed-forward layers with residual connection and normalization. However, there are also two significant differences to the encoder. The first attention layer is masked (see Section 2.3.2), which in this context means that only previous embeddings are taken into account in the attention mechanism. A second attention layer uses cross-attention, i.e. it takes the Query and Key matrices from the encoder while using the Value matrix from the previous layer of the decoder. Through this, the whole context provided by the input can be taken into account in the decoder. Lastly, a feed-forward layer follows analogous to the encoder.

The final selection of the next token is based on a linear layer, the softmax function and a sampling strategy which are all described below (see Section 2.4.3).

Once the next token has been determined, the input embedding of the decoder is expanded with it, and the updated embedding runs through the decoder recursively until a termination condition is met. This method, known as autoregressive encoding, generates token for token and concatenates them into a text [Vas+17].

The architecture described is suitable for processing a full sequence by the encoder and simultaneously using the provided information to generate a sequence by the decoder (seq2seq).

2.4.2. Encoder

An encoder is an element that transforms raw data, such as text, into a vector embedding. BERT (Bidirectional Encoder Representations from Transformers) is probably the best-known representative of the encoder class, which is presented below based on the work of Devlin et al. [Dev+19].

The structure corresponds exactly to the left half of Figure 3, i.e. it consists of attention and feed-forward layers where normalization takes place after a residual connection. It should be emphasized that the attention layer is bidirectional, i.e. it considers both left-sided and right-sided words in the context.

An encoder can be used to answer questions or extract features. Its output is an embedding. The encoder is trained by randomly masking tokens in an input of individual sentences, which are then predicted by the encoder. The error of the prediction can then be calculated using a loss function and the weights adjusted accordingly. In another method of training, the model determines whether one sentence follows the next or not. The training data therefore consists of sentence pairs and information as to whether it is actually the next sentence or not [Dev+19].

2.4.3. Decoder

This section first examines the decoder architecture and then explains the process of next token selection using temperature scaling and various sampling strategies.

Architecture The origin of the pure decoder architecture goes back to the work by OpenAI and forms the basis for the Generative Pretrained Transformer (GPT) class [Rad+18]. Figure 4 shows its structure, which largely corresponds to the decoder from Figure 3. The main difference is that the pure decoder has no cross-attention due to the absence of the encoder and this layer is therefore omitted. In contrast to the encoder, the attention layers of the decoder work unidirectional, i.e. only consider left-sided tokens in the context [Rad+18]. In newer implementations, normalization is performed before instead of after each layer [Xio+20].

During training, the model predicts the next token of a given sequence. Depending on the success, the weights are adjusted. Typical applications of decoders are text generation and completion tasks [Rad+18].

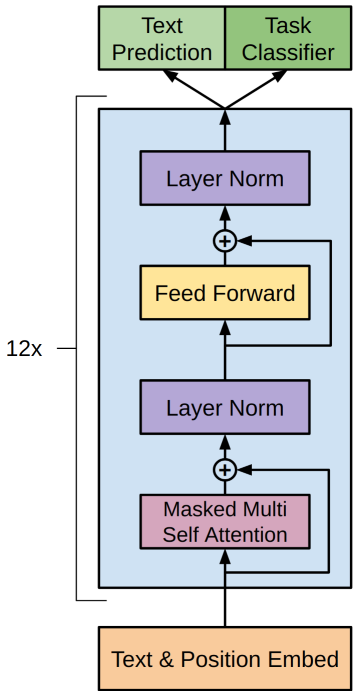


Figure 4: Architecture of a Decoder [Rad+18]

Next Token Selection The output of a decoder consists of vector embeddings. In this section, the final process of selecting the next token on basis of these embeddings is explained. It is inspired by the work of Minaee et al. [Min+25].

Temperature Scaling In a first step, a linear operation transforms the embeddings to a vector with dimension of the total vocabulary, i.e. all possible tokens. In a second step, the softmax function is used to calculate a probability distribution of these tokens analogous to Equation 20:

$$softmax(x_i) = \frac{e^{x_i/T}}{\sum_{j=1}^{n} e^{x_j/T}}, \tag{22}$$

but with an additional parameter T, referred to as temperature. To encourage more creativity, a flatter probability distribution can be generated by a temperature greater than one. Conversely, a temperature of less than one can lead to higher fact fidelity, as the probability distribution is sharpened. This technique is referred to as temperature scaling.

Sampling Strategies On the basis of the probability distribution obtained by the softmax function, different sampling strategies can be used to finally select the next token. The strategies below only represent a small selection of possible approaches.

- **Greedy Search:** The greedy search always opts for the most likely token.
- **Top-k:** With Top-k, the k most probable tokens are first preselected, where k is a predefined integer. The probabilities of the k tokens are normalized so that their sum is one. The next token is then chosen randomly based on the new probabilities.
- **Top-p:** With Top-p, the tokens are sorted by descending probability in a first step. Then the first tokens whose summed probability is greater than p are preselected. Analogous to Top-k, the probabilities of the remaining candidates are normalized and the next token is selected randomly.

2.5. Optimization and Extension of LLMs

Current LLMs are subject to the following limitations: (i) they are stateless (previous inputs and outputs are ignored), (ii) require resource-intensive training and inference, and (iii) are usually trained without domain-specific knowledge. In the context of this work, limitation (iii) is the most severe, as the lack of knowledge can lead to 'hallucinations', where LLMs produce fictional and unlearned outputs [Ama24]. One way to avoid this is to enrich LLMs with additional (domain-specific) knowledge using Parameter-Efficient Finetuning (see Section 2.5.1), Retrieval Augmented Generation (see Section 2.5.2) or Prompt Engineering (see Section 2.5.3).

2.5.1. Parameter-Efficient Fine-tuning

The knowledge of language models is stored in weights of matrices. After general training, a model can be specialized by further training with additional data. Adapting weights, particularly in large language models, is very computationally intensive and requires large amounts of training data. In order to integrate domain-specific knowledge more efficiently into a model, various methods are available, two of which are presented briefly. Figure 5 illustrates the architectures of the adapter technique [Hou+19] and Low Rank Adaptation (LoRA) [Hu+21].

In both methods, instead of adapting the already trained weights of a model, additional weight matrices are integrated, which are of lower dimension compared to the matrices of the model. During computation, the input is therefore projected to this lower dimension, the matrix multiplication is carried out, and then transformed back to the original dimension.

Both the adapter and LoRA are integrated into the environment of feed-forward or attention layers of an LLM. The adapter is connected serially behind the layer in front of the residual connection. In addition to the described transformation

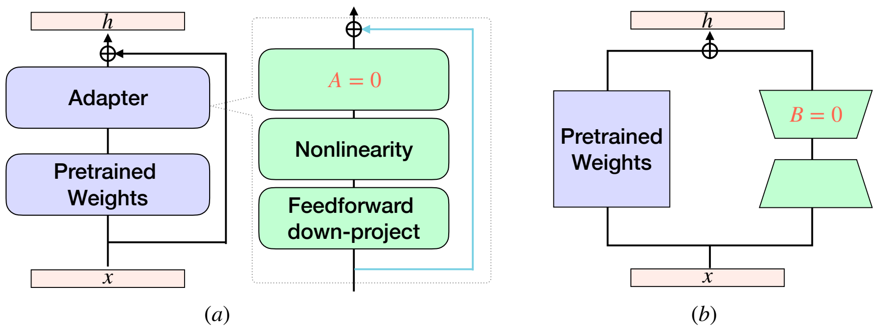


Figure 5: High-Level Architecture of a) Adapter and b) LoRA [Fu+22]

of the dimension, adapter calculations typically involve nonlinear transformations [Hou+19]. LoRA, on the other hand, is arranged parallel to the layer and consists of two matrix multiplications of matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, which contain the additional weights and simultaneously cause the dimension to be reduced and increased. The output of the layer $\boldsymbol{H}$ results from the input $\boldsymbol{X}$ by forming the sum of LoRA and the portion W_0 originally embedded in the model:

$$\boldsymbol{H} = \boldsymbol{W}_0\boldsymbol{X} + \boldsymbol{\Delta W X} = \boldsymbol{W}_0\boldsymbol{X} + \boldsymbol{BAX} \tag{23}$$

[Hu+21]. Both the adapter and LoRA are initialized so that they have little influence, which is illustrated by $\boldsymbol{A} = 0$ and $\boldsymbol{B} = 0$ in the figure.

2.5.2. Retrieval Augmented Generation

The following presentation is based on the work of Gheorghiu [Ghe24]. Retrieval

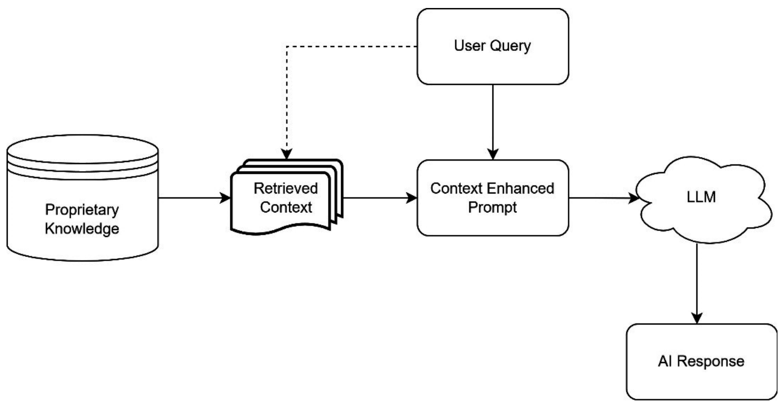


Figure 6: General principle of RAG [Ghe24]

Augmented Generation (RAG) is a framework that extends an already trained LLM with domain-specific knowledge. The weights of the model remain unchanged. Instead, the input of the LLM is extended by additional information from a database.

Figure 6 illustrates the RAG principle. The domain-specific knowledge is searched based on the query and relevant information is added to the input of the LLM. This task is performed by the so-called retriever. The generator, i.e. an LLM, takes these additional information into account when answering.

RAG can increase the quality of the answers while simultaneously reducing the probability of hallucinations. The latency of the system is increased by the retriever, but additional, computationally expensive training of the LLM is not necessary. RAG can be divided into classes according to the type of knowledge representation and the associated search in the data. The following paragraphs outline several key classes.

Dense Retrieval With dense retrieval, the knowledge data is converted into vector form by an encoder (see Section 2.2.2). The retriever finds relevant data using similarity measures between the vectorized query and the embeddings. For this purpose, methods such as Dense Passage Retrieval (DPR), FAISS and Approximate Nearest Neighbor (ANN) are used. The advantage of dense embedding is that semantic similarity can be detected while exact word match is not required. A disadvantage is the high computational effort when initially transferring the data into embeddings and the subsequent maintenance when changing the data [Ghe24].

Sparse Retrieval Sparse retrieval requires no pre-processing of the data and the method can be applied to large amounts of data. It searches for matches between the keywords of the query and the data and identifies the most important passages based on (relative) frequency (see Section 2.2.2). Sparse retrieval is characterized by simple traceability and works well on structured data such as books. The disadvantage of the method compared to dense embedding is that it cannot detect semantic similarity, but relies on exact word matching [Ghe24].

Knowledge Graph Instead of dense or sparse representations, a knowledge graph can be used to represent the database. In knowledge graphs, objects are represented by nodes and relationships by edges. The graph can be generated automatically from text documents using tools such as Stanford's OpenIE or BERT for Relation Extraction. The knowledge is stored in a structured form as triples (subject-predicate-object). Therefore, search is more explicit and as a result less unnecessary information is appended to the prompt [Che25].

Hybrid Approaches To increase accuracy, approaches can be combined to benefit from the strengths of different methods. For example, dense embedding can identify semantically similar terms in the query, which can then be found by sparse retrieval. Hybrid techniques are the subject of current research [WMC24].

2.5.3. Prompt Engineering

Prompt engineering (PE) is a method for improving the output quality of an LLM. The model itself and the learned weights remain unaffected. Instead, the idea is to refine the LLM's prompt in order to improve performance. Some basic techniques of prompt engineering are briefly outlined below, based on the work by Amatriain [Ama24].

Encouraging Factual Accurary A first possibility to improve the correctness of the output is to request factual orientation and source citations. The LLM is thus influenced not to hallucinate. To ensure that the LLM meets the user's requirements, it makes sense to use dominant and strict language. Even the punctuation plays a role in this.

The LLM can also be prompted to check itself. Once an answer has been generated, it can be passed again as input together with the original input and the question as to whether the answer is correct.

Few Shot Prompting LLMs have capabilities to process given examples and consider them to refine their output. This is the basis for a method referred to as few-shot prompting, in which the input is expanded with examples or additional information. This technique is therefore related to RAG with the difference that the examples provide information on format, while with RAG the prompt is fed external factual knowledge.

The order in which examples are given and questions are asked also affects the response. Tests have shown that examples should follow instructions and not vice versa.

Generation of Different Opinion In addition to the basic techniques mentioned, more complex methods are the subject of research. An increase in the quality of answers can be achieved by asking the LLM to formulate an opinion of different experts on a problem. This results in a multi-layered answer in which different perspectives are taken into account.

Chain Of Thought (COT) In Chain of Thought prompting, the LLM is asked to formulate a more complex train of thought in several steps instead of giving a direct answer.

Tree of Thought (TOT) Tree of Thought (TOT) prompting extends this concept. Instead of a chain, a tree of processing steps is created here. Each thought can therefore lead to several different subsequent thoughts. A final solution can then be synthesized by evaluating the individual thoughts [Ama24].

2.6. Summary

This section established the fundamentals of language processing, starting with the introduction to artificial neurons, with parameters (weights and biases) which influence the determination of the neuron's output (activation). For more complex tasks, several neurons can form a network and can be trained through backpropagation.

To adapt neural networks to language tasks, text must first be decomposed into atomic units, the tokens. These tokens can then be represented in vector spaces, known as embeddings.

The lexical approaches, e.g. BM25, use term frequency and other document statistics to measure term importance. They result in sparse, high-dimensional vector representations, where each entry is the frequency of a term, suitable for key-word based similarity search. Another approach is the transformer based vectorization, which captures semantics in lower-dimensional dense vectors. Both approaches are suitable to capture similarity between texts, while only the latter approach is used in further language processing.

Language models consist of feed-forward layers to learn complex relations and attention layers to take the context into consideration. Three main architectures are known – the encoder-decoder, encoder, decoder. To further enhance language models capabilities, they can be extended with Parameter-Efficient Fine-Tuning to learn new concepts, or with Retrieval Augmented Generation, which utilizes information from a database to provide more grounded answers. Another strategy is to use Prompt Engineering, which is based on specific prompt instruction techniques.

3. Fundamentals of Statistics on Ordinal Data

This section lays the foundation for the statistical analysis presented later in Section 6. It begins with a brief overview of ordinal data and statistical significance, followed by an introduction to relevant significance tests.

The Sections 3.2, 3.3, 3.4, and 3.5 are based on the work of Siegel [Sie56].

3.1. Ordinal Data

Ordinal scales categorize and rank data in a meaningful order, using for example variables like *often*, *sometimes* and *rarely*. These categorical variables can be assigned numbers (e.g. 1 for the lowest value *never*, 5 for the highest value *always*), but the key characteristic of ordinal scales is that the distances between two variables is neither equal nor meaningful.

Because the intervals are not consistent, parametric statistics with mean value and standard deviation are not appropriate. Instead, non-parametric statistics should be used to characterize a dataset, with measures like median and other quantiles as well as minimum and maximum values [Sve01].

3.2. Statistical Significance

When assessing study results, it is crucial to assure the findings are not just due to random chance, but represent the use case to be suitable for generalization. This is stated in the null hypothesis, which claims no difference. To reject this null hypothesis and thereby support the alternative hypothesis, tests were developed to be compared against a threshold significance level α. This level can be interpreted as the probability to reject a true null hypothesis (Type I error). The tests determine a p-value, which is the likeliness to observed the present sample if the null hypothesis was true. Therefore, the null hypothesis can be rejected, if the p-value falls below α.

The decision of the appropriate test depends on the exact boundary conditions. Siegel presents a taxonomy of statistical tests and when to use them [Sie56]. In case of ordinal scales, the subcategory of non parametric tests can be applied. In the study of this thesis, multiple configurations of the same system are tested on the same dataset (same questions) and therefore the samples are related, which further narrows down the suitable tests. For comparing exactly two samples the sign test or the more powerful Wilcoxon matched-pairs signed-ranks test can be applied. The latter one is only valid if the difference distribution of the two samples is roughly symmetrical around the median. For three or more samples, the Friedman two-way analysis of variance is suitable. The mentioned tests are briefly introduced in the following subsections on the basis of Siegel's work [Sie56]. It is followed by subsections regarding the reduction of the significance level α, if several tests are applied on the same dataset. The chapter finished with the Krippendorff's Alpha, which is a measure for the agreement of annotators.

3.3. Sign test

The sign test only uses the sign of the differences and neglects their magnitudes [DM46]. Apart from the aforementioned preconditions – ordinal data on related samples – there is no further condition which needs to be met.

The test presented is two-tailed, meaning there is no assumption about the direction of differences. The null hypothesis for this test therefore states that the probability of a positive and negative difference is both equal ($p = 0.5$).

The test procedure consists of the following steps:

1. Determine the sign of the score difference of each pair.
2. Ignore all pairs with zero differences (ties), leaving the remaining number of valid pairs N.
3. Sum the number of all positive and negative signs each. Select the smaller of the two as $k_{observed}$.
4. The p-value is the probability of encountering a result at least as extreme as k. Use the binomial probability Formula 24 to calculate the individual probabilities for $k = 0..k_{observed}$. The equation is:

   $$P(X = k) = \binom{n}{k} p^k (1 - p)^{n-k}, \tag{24}$$

   with $p = 0.5$ according to the null hypothesis and $n = N$.
5. To determine the final p-value, sum the individual probabilities and double the result to take the two-tailed distribution into account.

The sign test indicates if a significant difference in median values is encountered, but has no further implication on the magnitude.

3.4. Wilcoxon matched-pairs signed-ranks test

Compared to the sign test, the Wilcoxon matched-pairs signed-ranks test is a more powerful alternative, which takes not only the sign, but also the magnitude of differences into account [Wil45]. Historically, for larger sample sizes, an additional precondition of a roughly symmetrical difference distribution around the median was stated for justifying certain computational methods (see steps 6. and 7. below). With modern software, the exact p-value can be determined regardless the distribution and therefore this assumption is moot.

The test procedure for the two-tailed distribution consists of the following steps:

1. Determine the magnitude of the score difference of each pair.
2. Ignore all pairs with zero differences (ties), leaving the remaining number of valid pairs N.

3. Rank the pairs by their absolute differences, assigning the lowest rank 1 to the pair of smallest difference. If ties occur, assign the average rank.
4. Add the differences' signs to the ranks.
5. Sum all positive and negative ranks. Assign T the smaller of the two sums.
6. Convert T into a z-score if $N > 20$.
7. For $N > 20$, determine the final p-value by using the cumulative distribution function Φ:

$$\text{p-value} = 2\Phi(z), \qquad \text{for } z < 0 \tag{25}$$

$$\text{p-value} = 2(1 - \Phi(z)), \qquad \text{for } z \geq 0 \tag{26}$$

   For $N \leq 20$ use T and respective tables to determine the exact p-value.

Same as the sign test, the Wilcoxon matched-pairs signed-ranks test indicates a significant difference in median values. By adding the information of magnitude differences, the test is more powerful.

If the differences are significant, the effect size can be determined using the matched-pairs rank-biserial correlation. For this, the procedure of Wilcoxon matched-pairs signed-ranks test is slightly modified starting at step five:

5. Calculate the sums of all positive ranks R_+ and the sum of absolute values of negative ranks R_-.
6. The correlation can be determined using formula:

$$r = \frac{R_+ - R_-}{R_+ + R_-}, \tag{27}$$

   with values $-1 \leq r \leq 1$.

The rank-biserial correlation indicates the magnitude and direction of rank differences [Ker14]. The following reference values are given [IL24]:

- $0.1 \leq |r| < 0.3$: small effect,
- $0.3 \leq |r| < 0.5$: medium effect, and
- $0.5 \leq |r|$: large effect.

3.5. Friedman two-way analysis of variance

The Friedman two-way analysis of variance is suitable to compare more than two related sample groups to examine the statistical significance of difference [Fri37]. The null hypothesis states that all median values are the same over all groups.

The test procedure for k groups is as follows:

1. For each sample, rank the results over each group from 1 to k.
2. Sum the ranks of each group j: R_j.
3. Determine χ_r^2 using formula:

$$\chi_r^2 = \frac{12}{Nk(k+1)} \sum_{j=1}^{k} R_j^2 - 3N(k+1), \tag{28}$$

   with samples size N.
4. The p-value is determined on the basis of the chi-square distribution and the degree of freedom $k-1$ using respective tables or software.

If the Friedman test implies significant differences over all samples, the sign or Wilcoxon test can be applied pairwise to determine which of the samples actually differ. Since multiple tests are applied and their p-values are compared to a static significance level α, the likeliness of a type I error increases. This issue is addressed in the next section using the Holm-Bonferroni correction.

3.6. Holm-Bonferroni

If multiple hypothesis are tested on the same data set, the risk of rejecting a true null hypothesis (type I error) increases with each test. To compensate, the significance level α must be decreased to set a stricter threshold. The Holm-Bonferroni method is one of several techniques for this, decreasing in a step-down procedure [Hol79]:

For examining n hypothesis, sort their respective p-values by magnitude in ascending order. Starting with the first p-value, compare to the decreased significance level α/n. If the p-value is below this threshold, continue with the second p-value and compare it to $\alpha/(n-1)$. Continue until all p-values are processed or a p-value exceeds the adapted threshold. In the later case, stop the comparisons and do not reject the hypothesis regarding this p-value and subsequent (larger) p-values.

The sign-, Wilcoxon and Friedman test, complemented with the Holm-Bonferroni method, can be used to evaluate statistical differences in samples. To further validate the results, it is crucial to assess the quality of study design. The following section discusses Krippendorff's Alpha, a measure for annotator agreement, as quality indicator.

3.7. Krippendorff's Alpha

Krippendorff's Alpha is a measure for agreement of annotators or, more generally, observers [Kri19]. It can be flexibly applied to datasets regardless of sample size, number of observers, type of scale and is robust against missing data. For ordinal data, it is defined as:

$$\alpha = 1 - \frac{D_o}{D_e}, \tag{29}$$

with observed disagreement D_o:

$$D_o = \frac{1}{n} \sum_c \sum_k o_{ck} \delta_{ck}^2, \tag{30}$$

and expected disagreement D_e, i.e. if values would be assigned by pure chance:

$$D_e = \frac{1}{n(n-1)} \sum_c n_c \sum_k n_k \delta_{ck}^2. \tag{31}$$

The total number of pairable units n, the observed disagreements o_{ck}, the disagreement weight δ_{ck} and the total number of ratings per category n_c and n_k are determined in these steps:

1. Create a reliability data matrix, with each unit of the sample as one column and one observer per row. Covert categorical labels to numerical ranks and insert them into the matrix.

2. Add an additional row which counts the number of assessments per unit m_u. If no data is missing, it's equal to the number of observers.

3. Create a coincidences matrix with cell values:

$$o_{ck} = \sum_u \frac{\text{Number of c,k-pairs in unit u}}{m_u - 1}. \tag{32}$$

4. Count the frequency of each rank n_c with $1 \leq c \leq rank_{max}$.

5. Determine ordinal metric differences, which assign rank differences a weight, with formula:

$$\delta_{ck}^2 = (\sum_{g=c}^{g=k} n_g - \frac{n_c + n_k}{2})^2, \tag{33}$$

   for $1 \leq c, k \leq rank_{max}$ [Kri19].

For the determined Krippendorff's Alpha α, reference values are given for comparison [MBM24]:

- $\alpha = 1$: perfect agreement,
- $\alpha \geq 0.80$: indicating reliable rating,
- $0.67 \geq \alpha \geq 0.79$: moderate reliability,
- $\alpha < 0.67$: poor agreement and therefore unreliable data,
- $\alpha = 0$: no agreement, and
- $\alpha < 0$: systematic disagreement.

3.8. Summary

This section on statistics for ordinal data explained the use of non-parametric methods. It introduced several tests to examine the reliability of findings. The Friedman two-way analysis of variance can be applied to check for statistical significance when comparing more than two datasets, while the Sign or Wilcoxon matched-pairs signed-ranks test can follow to distinguish which of these datasets actually differ. If multiple tests are run on the same family of datasets, the Holm-Bonferroni correction must be considered to avoid type I errors. In addition, Krippendorff's Alpha further investigates reliability by examining annotator agreement rather than statistical significance.

4. Related Work

In the following subsection, related studies on RAG are reviewed as a point of reference and inspiration for this thesis. It is followed by a subsection dedicated to VLM benchmarks. Subsequently, evaluation metrics for generated text are compared.

4.1. Related research

LLMs show strong performance on several tasks like summarization, translation and question answering. A major unsolved challenge described in the literature is hallucinations. This occurs when a model generates factually inaccurate content, which seems plausible at first glance [Hua+25]. One reason for this unreliability issue is a model facing an unknown domain it was not trained on. As a countermeasure, Retrieval Augmented Generation (RAG) was developed (see Section 2.5.2). This approach enhances the model by providing additional context from a database to the prompt. However, the main issue with this promising method is to find the highly relevant information to the query, while irrelevant and noisy context lowers performance.

Pure text-based RAG has been extensively explored in multiple studies on several different domains and tasks [Gao+24]. In contrast, approaches utilizing information from text and images are still less examined.

Covered Domains and Datasets Tailoring a RAG system might introduce different challenges depending on the domain, as some concepts or terminology might be more rare and therefore harder to process. When reviewing the domains which were covered in RAG-related research, it demonstrated a wide range of use cases. Research focused on question answering on general test datasets [Che+22; LB22], nature [Wan+25], science [Yag23], retail [Thi25] and medicine [WMC24; Shi+25]. Moreover, the technical sphere was explored in telecommunications [Xio+25], software desktop applications [Sha+24] and industrial programmable controllers [RL24].

For evaluation of their RAG approaches, most researchers rely on synthetic questions [Che+22; LB22; Wan+25; Yag23; WMC24; Shi+25; Xio+25; RL24], while fewer studies utilize real customer requests [Sha+24]. While synthetic questions enable comparing different implementations of components, they do not reveal the real-world performance, where questions may be vague or noisy. Therefore, it remains unknown whether these RAG applications are deployment ready.

The domain investigated in this thesis is about software for programmable controllers. While research on a similar domain exists [RL24], this thesis uses real customer requests to examine performance, instead of synthetic questions. It aims to contribute to the research gap in the industrial domain evaluated on a real-world test set, which may reveal additional challenges.

Retrieval Implementation While some retrieval implementations are limited to only text [WMC24; Yag23; Wan+25], multimodal approaches utilizing text and images are explored, which differ in their exact implementation. The main difference is how the retrieval step is conducted, whereas all of them exploit a VLM for the final generation step.

In a text-only approach, retrieval based on BM25 (sparse) and embeddings based on encoding (dense) both showed strong performance on well formulated questions [Wan+25]. Evaluating on reformulated questions of identical semantics, the dense technique excels compared to its counterpart, while the sparse approach still demonstrated merit. Therefore, both methods are successfully combined in hybrid retrieval [WMC24].

When facing multimodal datasets, different implementations regarding handling the modalities are explored. Some studies depend on image summaries for retrieval, which are previously generated by VLMs. This approach allows storing text and image representations in a database which exclusively consists of text-only embeddings [RL24; Thi25; Xio+25]. All these approaches have in common that they only exploit dense retrieval techniques, even though the text-only approach would allow for sparse techniques as well.

In one study, the image-summary approach was compared to using specialized encoders for each modality and therefore utilizing two separate databases. Neither approach could significantly outperform the other [RL24].

Another implementation uses an encoder, which embeds text and image into a shared vector space. This newer approach is examined in [Che+22] but needs further investigation [Zha+23]. As more ready-to-use encoders of this type are developed (see Appendix A.1), this approach offers straightforward implementation.

All the reviewed studies have in common that none of them explains why the respective implementation was chosen over others, leaving the understanding of design decisions to speculation. As each approach has trade-offs, it remains unclear whether one of them is superior. However, it is noticeable that none of the multimodal approaches leverages sparse and dense techniques, even though in text-only approaches it was observed to be beneficial. This gap in hybrid multimodal retrieval presents an opportunity for further investigation.

Other Implementation Details Besides different retrieval strategies, the reviewed RAG approaches differ in implementation details and focus on different aspects.

Two studies indicate that model generators of smaller size can perform competitively with significantly larger models [Xio+25; RL24]. Wang, Ma, and Chen examined different enhancing methods for RAG and revealed their individual impact in an ablation study (see Table 3). By adding a retriever, the benchmark scores were increased by up to 14% (depending on the dataset) compared to the plain model without utilizing RAG. Rewriting the query to increase preciseness in the subsequent retrieval step showed improvements of roughly 6%. Additionally, the knowledge self-refiner after retrieval, which filtered the retrieved chunks, led only to a slight

Method	MedQA-USMLE	MedQA-MCMLE	Med-MCQA
GPT-3.5-Turbo	51.3	58.2	53.9
+ retriever	58.6	61.2	57.1
+ retriever + augmented query	62.0	65.4	63.1
+ retriever + knowledge self-refiner	63.9	68.1	64.4
+ retriever + augmented query + knowledge self-refiner	65.0	68.8	65.1
+ fine-tuned retriever	61.2	62.3	58.7
+ fine-tuned retriever + augmented query	64.1	68.9	63.4
+ fine-tuned retriever + knowledge self-refiner	65.7	70.3	64.8
+ fine-tuned retriever + augmented query + knowledge self-refiner	67.9	72.6	65.5

Table 3: Ablation study of different components examined on the medical data sets MedQA-USMLE, MedQA-MCMLE and Med-MCQA [WMC24].

increase in performance (around 1%). Fine-tuning the retriever and applying both query augmentation and knowledge self-refining further increased performance by 1% to 6%, compared to the same setup without fine-tuning. This indicates that the biggest performance enhancer is the retriever itself, which can be complemented by additional measures showing potential for further improvements.

In another study, retriever performance was compared to a fine-tuned variant. Among the unmodified retrievers, the performance differences were huge (e.g. scores of 0.70 compared to 0.16). At the same time, the best performing retrievers showed competitiveness compared to the fine-tuned version (0.72 vs. 0.73). While fine-tuning has value, the improvements appear rather small in relation to the expense. These findings suggest that carefully selecting a retriever might be more cost-effective than fine-tuning a mediocre one.

4.2. Vision Language Model Benchmarks

The performance of various LLMs and VLMs is evaluated using standardized tests. These usually consist of a list of questions and answers covering various topics such as art, technology, etc. A wide range of skills are tested, from summarizing texts and logical thinking to letter and word recognition in images.

While some benchmarks cover as many skills and/or subjects as possible, others are much more specialized. The main features of the benchmarks relevant to this work are presented below.

DocVQA DocVQA is a VLM test and comprises a question catalog created on the basis of industrial documents. The task is to extract information from images showing document pages with text, tables, lists and other illustrations, where the focus is on understanding the layout. Additionally, some questions address recognizing handwriting, photography and other topics [MKJ21].

MMMU MMMU (Massive Multi-discipline Multimodal Understanding and Reasoning Benchmark) tests six different subject areas: Art and Design, Business, Science, Health and Medicine, Humanities and Social Sciences, as well as Technology and Engineering. Knowledge and reasoning are tested at university level by understanding diagrams, maps, tables etc., extracting information and drawing conclusions. There are both free-response and multiple-choice questions [Yue+24].

AI2D AI2D contains 15000 questions on scientific diagrams at primary school level. The VLMs have to understand the diagrams and answer questions about them, some of which require simple logical reasoning [Kem+16].

OCRBench OCRBench (Optical Character Recognition) focuses on recognizing text in images, which are divided into five task types. In addition to the text recognition of hard-to-read fonts and handwriting, text information must be extracted from photos (e.g. street signs). Furthermore, information must be extracted from graphics and documents. All questions are answered in free form [Liu+24b].

MMBench Multimodal Bench contains over 3000 multiple-choice questions and, according to the authors, covers 20 skills. These include interpreting the subject of a picture, recognizing writing, identifying well-known personalities and many more. The special feature of the evaluation is that the questions are asked several times and the answer options are swapped each time. Only if the VLM gives the correct answer in all cases is the question considered to have been passed [Liu+24a].

The selected benchmarks form the foundation for selecting VLMs (see Section 5.1.3 and Appendix A.2), which are then further evaluated in a feasibility study as part of a RAG pipeline.

4.3. Evaluation Metrics

The following section provides a brief overview of evaluation metrics of Natural Language Generation (NLG) and discusses advantages, disadvantages and particularities of each approach.

The evaluation of text generation is a complex task that requires careful design and preparation. According to Celikyilmaz, Clark, and Gao, evaluation metrics can be categorized into automatic metrics, machine-learned metrics, and human evaluation [CCG21].

Automatic Metrics Automatic metrics compare a reference text (ground truth) with a computer generated text in a predefined way. Overlap metrics like BLEU or ROUGE measure how many n-gram (1-word, 2-words, n-words) match within the two texts to compare. Looking at the example 'I write a thesis' as a reference, the sentence consists of the four 1-grams 'I', 'write', 'a', 'thesis', the three 2-grams 'I write', 'write a', 'a thesis' and so on. A candidate sentence 'You write a poem' therefore matches in the two 1-grams 'write' and 'a', as well as the 2-gram 'write a'. The final similarity score is then computed on the proportion of matching n-grams compared to the total n-grams. The Edit distance is a distance-based metric, which scores how many substitutions, insertions and deletions of characters are necessary to equal two texts. To transform 'kitten' to 'sitting' requires two substitutions ('k' to 's', and 'e' to 'i') and one insertion ('g'). The number of operations builds the final distance score.

Machine-learned Metrics Another category in NLG evaluation is machine-learned metrics, which use language models to assess semantic similarity. BERT-score is a representative of the machine-learned metrics. It uses a pretrained Encoder (see Section 2.4) to transform the text into dense vector embeddings and calculate the cosine similarity between them. Compared to automatic metrics, similarity in semantics can be captured. BERT-score could therefore recognize a high similarity between the reference phrase "stop at the waterfront" and the candidate phrase "they halted at the shore", whereas overlap metrics would miss out on it.

Human Evaluation Human evaluation of NLG can be either extrinsic or intrinsic. The extrinsic approach checks the success in a downstream task of the system itself or its user. For example, if patient information is generated and given to doctors, the evaluation would measure the quality of diagnosis these doctors provide. In contrast, intrinsic evaluation focuses on the quality of the generated text itself. Most common approach is to use rating scales on predefined criteria like accuracy, relevance, or fluency. Another method is to compare generated texts to each other and rank them. Using rating scales, the absolute quality can be measured, while with ranking only relative quality is obtained [CCG21].

Human must be performed meticulously and requires proper planning. The goals of the study should be explicitly defined and if rating scales are used, the criteria should be chosen accordingly [HA21]. When designing the questionnaire, these criteria should not only be named, but also explicitly and clearly defined. Howcroft et al. reviewed several NLG evaluation studies noticing a lack of definitions. They also noticed that several terms are used to describe the same measured aspect and vice versa, the same terms were used to actually describe different aspects. This leads to confusion when comparing studies among each other and reduces the quality of assessment, because annotators may interpret the criteria differently [How+20].

Among annotators, there might be unconscious bias [RWW24] and experts usually judge stricter than novices in domain specific tasks [AC24]. Therefore, best practice is the assessment with at least three annotators per sample to ensure statistical reliability [van+21]. In general, if the annotator's agreement is not well aligned, it indicates poor test design. However, in NLG taks, the agreement is usually poor due to variety of language [CCG21].

While the questionnaire must be unambiguous, it should not be too rigid at the same time. If annotators feel too restricted, edge cases might not be handled properly [RWW24]. For example, if text generation fails entirely due to technical problems, it might be more preferable to allow skipping the item with a short justification. Otherwise, there is the danger that annotators assign high scores, because the blank answer was, for example, neither harmful nor were any hallucinations identified.

During the annotation phase, the workload of each annotator should be planned well to prevent fatigue. When assessing text of multiple models, especially when compared to each other, the presentation order should be randomized to prevent bias.

LLM-as-a-Judge In addition to the three categories of metrics presented above, current state of research is the use of an LLM-as-a-Judge. In this approach, the LLM follows instructions similar to human annotators [AC24]. It can be seen as a mix of machine-based and human evaluation, trying to combine their strengths.

Compared to human evaluation and analogous to machine-based metrics, LLM-as-a-judge is highly scalable. To ensure quality of assessment, the LLM used for evaluation must be either more capable (higher-order judge) compared to the one generating the text, or must have been evaluated beforehand [AC24]. In a comparison of experts and an LLM as annotators on expert knowledge tasks a match of approximately 64% was observed [Szy+25]. Li et al. identified several biases and vulnerabilities regarding LLM as judges. LLMs rate their own answers very high, showing an egocentric bias. Also, the order in which texts are presented to LLMs influences their opinion on them (positional bias). Furthermore, a verbosity bias was observed, where LLMs favor either very short or long answers regardless of quality. A bias towards authorities favors content received from those. Additionally, LLMs are vulnerable to sentiment, favoring positive formulation [Li+25].

Analogous to human questionnaires, the criteria and prompt for guiding the judging LLM should be prepared carefully.

Discussion Celikyilmaz, Clark, and Gao and Sai, Mohankumar, and Khapra provide broad overviews of the landscape of automatic and machine-learned metrics, which need to be selected carefully depending on the use-case. One major advantage they all have in common is their scalability making them suitable for huge sample sizes. Human evaluation on the other hand is very cost intensive, but is still considered the gold standard regarding the quality of assessment. Sai, Mohankumar, and Khapra gather and discuss limitations of machine-based metrics. These metrics generally correlate poorly with human evaluation: BLEU and other overlap metrics even correlate negatively on fluency and showed low correlation on adequacy. Additionally, the machine-based assessment typically generates one single aggregate score, which cannot be further interpreted regarding different aspects such as fluency, accuracy, or factuality. This criticism goes hand in hand with poor adaptability to specific tasks with possibly very different requirements. Moreover, even task-specific metrics cannot capture all nuances reliably [SMK22] [van+21].

In summary, automatic metrics and LLM-as-a-Judge offer great scalability, while human evaluation provides the most distinct assessments, if conducted carefully. Even though it is labor intensive, it is suitable for the feasibility study of this thesis (see Section 5), as several experts are available for annotation.

5. Methodology

This section outlines the methodology of the case study conducted in this thesis. Six different RAG configurations are tested on a proprietary dataset and their answers are annotated by domain experts.

The section begins with a description of the design of the evaluated RAG system. Implementation details on the preprocessor component, the retriever, and the generator are presented, including a section on how the generator models for this study were preselected. Subsequently, more information on the study execution and annotation process is provided.

5.1. Design

The RAG system is structured into four major components (see Figure 7), each deployed as an isolated Docker container[3]. The user interacts with the system only through the composer, which orchestrates the RAG process by calling the preprocessor (see Section 5.1.1), the retriever (see Section 5.1.2), and the generator (see Section 5.1.3).

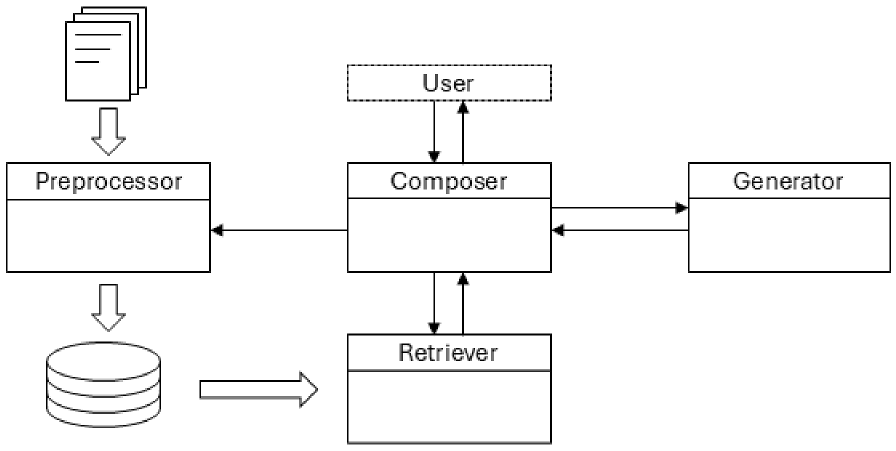


Figure 7: System architecture of the proposed RAG pipeline

Using one docker container per component isolates dependencies. As multiple libraries are used – sometimes cross-dependent – and CUDA[4] support is needed for GPU acceleration, separating concerns significantly reduces complexity. This approach also increases maintainability and extensibility, as changes on one component do not affect others. The disadvantage is additional HTTP communication overhead between the isolated components. However, this overhead is easily man-

[3]https://www.docker.com/resources/what-container/
[4]https://developer.nvidia.com/cuda-toolkit

aged with appropriate libraries and minimally affects inference speed, as the retriever and generator are the computational bottlenecks. Therefore, the benefits of maintainability and dependency isolation outweigh the additional communication costs.

5.1.1. Preprocessor

The preprocessor is called only once for initialization or when new documents are added to the database. It processes the database, which consists of manuals and other technical descriptions as well as C-code software documentation in form of header files, to generate chunks.

For chunking the PDF manuals, Python's Unstructured library[5] is utilized to extract the elements like section titles, text, images etc. On this basis, processing steps are applied (see Figure 8). Firstly, the headers and footers of the documents are ignored, because they would introduce noise. Subsequently, chunks are built hierarchically at the section level, cutting down to the third-tier subsections. This strategy takes advantage of the manuals' given structure, which in itself provides self-contained, independent pieces of information. Because sparse representations only work on text, descriptions of extracted images are generated by a VLM (gemma-3-27b-it-Q2_K_L) and added to the respective chunks to enable BM25 (see Section 5.1.2) to process the images' information.

All header files of the software documentation comply to the same pattern regarding structure: the documents start with a general description of the subject, e.g. a software module, followed by the interface, which mainly consists of function prototypes with respective descriptions. The introduction part and each function header is represented in a chunk.

With these chunking approaches, the chunks can vary in length and also modality, with some containing only text, while others include images and text.

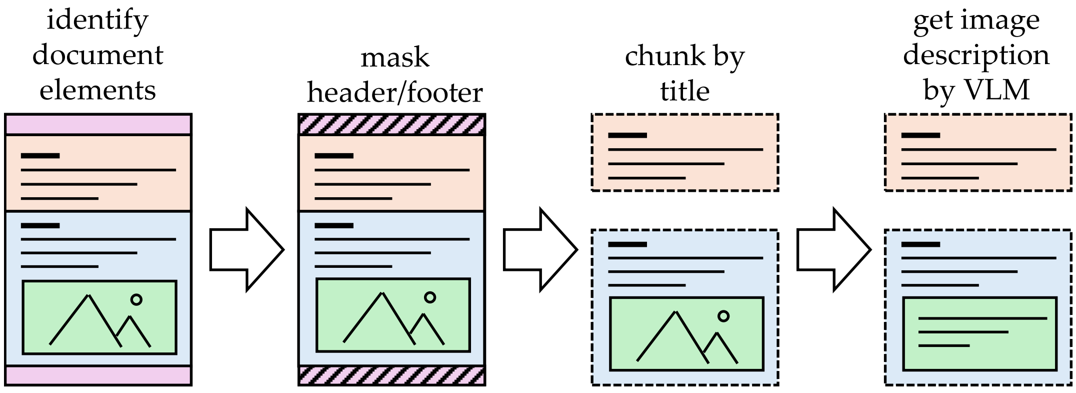


Figure 8: Preprocessing steps of documents

[5]`https://pypi.org/project/unstructured/`

5.1.2. Retriever

The retriever consists of three sub-components for sparse, dense and hybrid retrieval.

Sparse For the sparse component, all chunks and the query are initially lowercased and stemmed. Then, the BM25 algorithm (see Section 2.2.2) is applied to find the best matching chunks. Like mentioned in the previous subsection, the images are represented by their descriptions. Therefore, if a query contains an image, a description is initially generated by the generator VLM before applying the algorithm.

Dense The dense component utilizes ChromaDB[6] as a database for storing the embeddings. Regarding the encoding strategy, several approaches are possible:

a) Encode text and image simultaneously (early fusion).

b) Encode text and image separately and fuse afterwards (late fusion).

c) Encode text and image separately and don't fuse them. Instead, let both point to the same chunk.

Variant a) associates both modalities early and thereby enabling the capture of semantic relations. In contrast, variant b) encodes both modalities separately, which may result in some information loss. For variant c), the same issues as for b) apply, and an additional degree of freedom is introduced when computing the final similarity score of a chunk that is represented by multiple embeddings.

Under the assumption that the technical documentation's figures are referenced and explained in the text or are otherwise closely related, variant a) is chosen in this context. Thus, the encoder model must have the ability to build one encoding for simultaneous text and image input. Furthermore, the context window should be sufficiently long, to handle full subsections of the manuals. To identify a feasible model, the properties of multiple encoders were compared (see Appendix A.1), leading to the selection of Ops-MM-embedding-v1-2B[7].

For computing similarity between a query and the chunks, the Euclidean distance is used, which is the standard setting in ChromaDB.

Hybrid The hybrid approach combines sparse and dense retrieval by first calling these sub-processes (see Figure 9). Subsequently, the retrieved chunks of both strategies are merged into one, following the procedure described by Bruch, Gai, and Ingber [BGI23] and Shah et al. [Sha+25]. The BM25 algorithm returns similarity scores, meaning higher values indicating better matches. Contrarily, the dense retriever outputs distances, with an inverse relationship, where lower values indicate better

[6]`https://www.trychroma.com/`

[7]`https://huggingface.co/OpenSearch-AI/Ops-MM-embedding-v1-2B`

matches. Therefore, the reciprocal of the distances is used to translate into similarity scores. For the edge case of zero distance, a fixed, high-similarity value is assigned. Even though both scales now represent similarity values, their magnitude will be far off, making them unsuitable for comparison yet. Thus, both scores are normalized with standard score for final comparison and combination:

$$z = \frac{x - \mu}{\sigma}, \tag{34}$$

where x is the value to normalize, μ is the mean value and σ is the standard deviation. In a final step, the normalized sparse and dense scores are averaged for each chunk. This favors chunks which are retrieved by both methods, whereas chunks retrieved by only one are assigned a zero value for the non retrieved strategy.

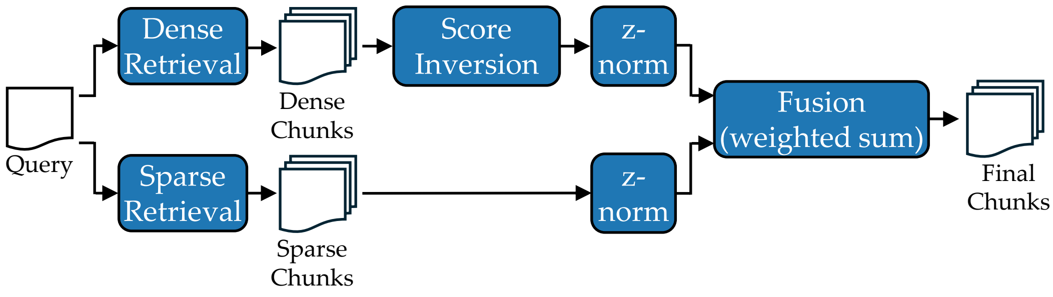


Figure 9: Hybrid retrieval architecture featuring dense and sparse retrieval in parallel branches. The dense distance scores are converted into similarity scores by inversion before both the dense and sparse scores undergo z-normalization. The final rankings are determined through score fusion in a weighted sum.

To ensure a statistically robust calculation of the standard score, a larger number of chunks is initially retrieved by sparse and dense approach. After merging the scores, only the desired number of highest ranked chunks is returned as a final result. Regarding normalization, the standard score is chosen, because it is more robust against outliers compared to the min-max normalization. This is because the mean and standard deviation is computed over the whole population, making it less susceptible to outliers compared to single values like minimum and maximum.

5.1.3. Generator

In the multimodal, text-image RAG pipeline, the generator is a VLM responsible for answering the user query based on the retrieved content. In this thesis, the generator is also requested to describe images for the sparse retrieval approach (see Sections 5.1.1 and 5.1.2).

This study examines three models and one quantized variant. Their selection is briefly outlined here (for the complete process see Appendix A.2). In an initial preselection step, a literature review of VLMs was conducted to identify candidates compatible with the computational constraints of this study. Subsequently, two benchmarks were selected to evaluate performance on the crucial tasks of the generator in this RAG pipeline, specifically logical reasoning, the interpretation of tables, and text recognition in high-quality structured documents. A total score was calculated as the mean of these two benchmark scores for each model. The final ranking was based on this score, resulting in the following selection of models:

- **InternVL3** (InternVL3-8B-Instruct[8])
- **Phi-4** (Phi-4-multimodal-instruct[9])
- **Gemma3-4B** (gemma-3-4b-it[10])
- **Gemma3-27B** (gemma-3-27b-it-Q2_K_L[11]).

Note that the shortened names written in bold are used from here on for easier readability.

Due to availability in Gpt-Generated Unified Format (GGUF), Gemma3-27B and InternVL3 are run using llama-cpp, while Phi-4 and Gemma3-4B are utilized by huggingface's transformer library.

For the generation of the final answer, a prompt template is used, which defines a specific role, provides instructions to the task, gives a brief description of the environment, and includes an example. For the whole template see Appendix A.3.

5.2. Experimental Setup

In the experiments, six different RAG configurations are examined on 65 real customer questions, which were selected on the basis that they are content-related and answerable with regards to the database. The answers are assessed by domain experts and the questions manually anonymized beforehand; however, spelling errors or wrong choice of words were not corrected. For the annotation process, each question is assigned a ground truth answer, which briefly outlines the correct technical response providing sufficient information for a domain expert.

The research questions focus on the answer quality of the different RAG setups by comparing generator models to each other as well as different retrieval strategies. Methodologically, it would make sense to determine the best performing retrieval strategy first and subsequently fixate it when comparing the generator models. Due to time constraints, a more concise approach is taken: Each model is paired with the hybrid retrieval strategy, because it is expected to provide the most precise results by combining the strengths of dense and sparse retrieval [MM24]. For the

[8]https://huggingface.co/OpenGVLab/InternVL3-8B
[9]https://huggingface.co/microsoft/Phi-4-multimodal-instruct
[10]https://huggingface.co/google/gemma-3-4b-it
[11]https://huggingface.co/unsloth/gemma-3-27b-it-GGUF

retrieval comparison, one model must be chosen and tested with dense-only and sparse-only retrieval to isolate the impact of each approach. Given that the benchmarks for InternVL3, Gemma3-27B, and Phi-4 are comparable (see Appendix A.2), the parameter count is used as a tiebreaker to maximize model capability. Therefore, Gemma3-27B is selected.

Retrieval computations were performed on CPU, while the GPU was dedicated to the generator. In rare cases of exceptional long contexts, the generator needed to be (partly) offloaded to CPU due to growing kv-cache.

5.2.1. Annotation Procedure

The annotation of the generated answers was conducted by eleven domain experts. To mitigate bias, each answer was reviewed by three different people. With six RAG setups and 65 questions, 390 answers were annotated. The author of this thesis annotated each answer once. The other ten experts were assigned 78 answers each, resulting in the required three annotations per answer.

For the annotation, an Argilla[12] server was deployed on huggingface spaces. Among other features, Argilla provides convenient management of annotation processes, including assigning datasets to specific groups and designing custom questionnaires (see Section 5.2.2 and Appendix A.6).

To accommodate the annotator's schedules and other responsibilities, the annotation process was rolled out over two weeks. This flexibility allowed experts to work in their preferred environments, e.g. when working from home in a quit setting. This also lowers the likelihood of fatigue, as pauses can be taken at individual demand.

Preceding the annotation start, a briefing with further instructions took place:

- The ground truth answer serves as a guideline, but might not be the only acceptable answer. Therefore, the experts knowledge is prioritized over the ground truth.
- Only the technical aspects of the answer were to be accessed, while for example salutations were to be ignored.
- Experts were instructed to order the questions by ID, meaning experts will see a question six times in a row with different answers of the examined setups. This minimizes cognitive load and improves consistency, because of familiarity with the question.
- Experts were instructed to be precise when accessing each annotation category. Two examples were given for demonstration.
- Experts were required to read the additional explanation of each category prior to the annotation.

[12] `https://argilla.io/`

- Experts were encouraged to add an additional comment, if difficulties are encountered, for example, if the answer does not match the question indicating an error in the dataset.
- Experts were instructed to immediately contact the author of this thesis in case of questions instead of proceeding based on false assumptions.

5.2.2. Questionnaire Design

The design of the questionnaire must be tailored to the purpose of the survey. In this thesis, only the answer quality is to be assessed, while criteria like inference speed, for example, is ignored. Also, the system boundaries are defined as the question-answer pairs and intermediate states like retrieved chunks are not provided to the annotators.

To balance annotation effort and precision, four key quality metrics were chosen on the basis of frequently used categories in the literature [van+19; How+20]. As Howcroft et al. point out, category names are used differently across studies while trying to measure the same aspect and therefore, the explicit definition of each category is key [How+20].

The following four categories are selected for this study to address the regarded questions:

- **Helpfulness and Completeness:** Does the provided answer contain information which solve the problem described in the request? If so, to which extent?
- **Correctness and Factual Accuracy:** To which extent is the provided information objectively correct?
- **Clarity and Fluency:** Is the provided text easily comprehendable with regards to structure and formulations? If so, to which extent?
- **Conciseness:** To which extent is the text free from repetitions, unnecessary verbosity and passages of low informational value?

The first quality measurement is the general *Helpfulness*. This is combined with *Completeness*, because there is a high correlation between the two: An answer is particularly helpful, if it provides all of the required information.

The second category, *Correctness and Factual Accuracy*, is distinct from the first, because an answer can be factually correct, but not relevant or helpful. Vice versa, it can provide information on why an answer might not be helpful, for example, if it is less factual.

Clarity and *Fluency* are combined into a single category to avoid unnecessary complexity for the annotators, as these categories are highly correlated.

The categories are complemented with *Conciseness* to measure if the answer provides only the required information or adds unwanted noise.

The categories are evaluated on a 5-point ordinal scale. The differentiation of five scores is recommended in the literature [AT22] and most widely used in automatically generated text [van+19]. Compared to 3-point or 7-point scales, the 5-point scale provides a good compromise of high reliability and ease of responding process for the annotators.

To ensure precision and consistency among annotators, each score value is defined by an additional explanation for more distinct differentiation.

5.3. Summary

In a feasibility study, six RAG configurations are compared on a test set of 65 anonymized real-world requests. The database is built on the basis of software manuals and documentation, which is chunked with a hierarchical approach using the section headers.

Three retrieval strategies are compared using an encoder's dense embeddings, the sparse BM25 algorithm, and a hybrid approach, which combines both approaches. Moreover, three generator models and a quantized version of different parameter count are examined. Each answer of the RAG pipeline is assessed by three domain experts using a 5-point ordinal scale on four categories.

6. Experiments

This section presents the study's results. It starts with an overview, continues with statistical calculations, and ends on annotator agreement.

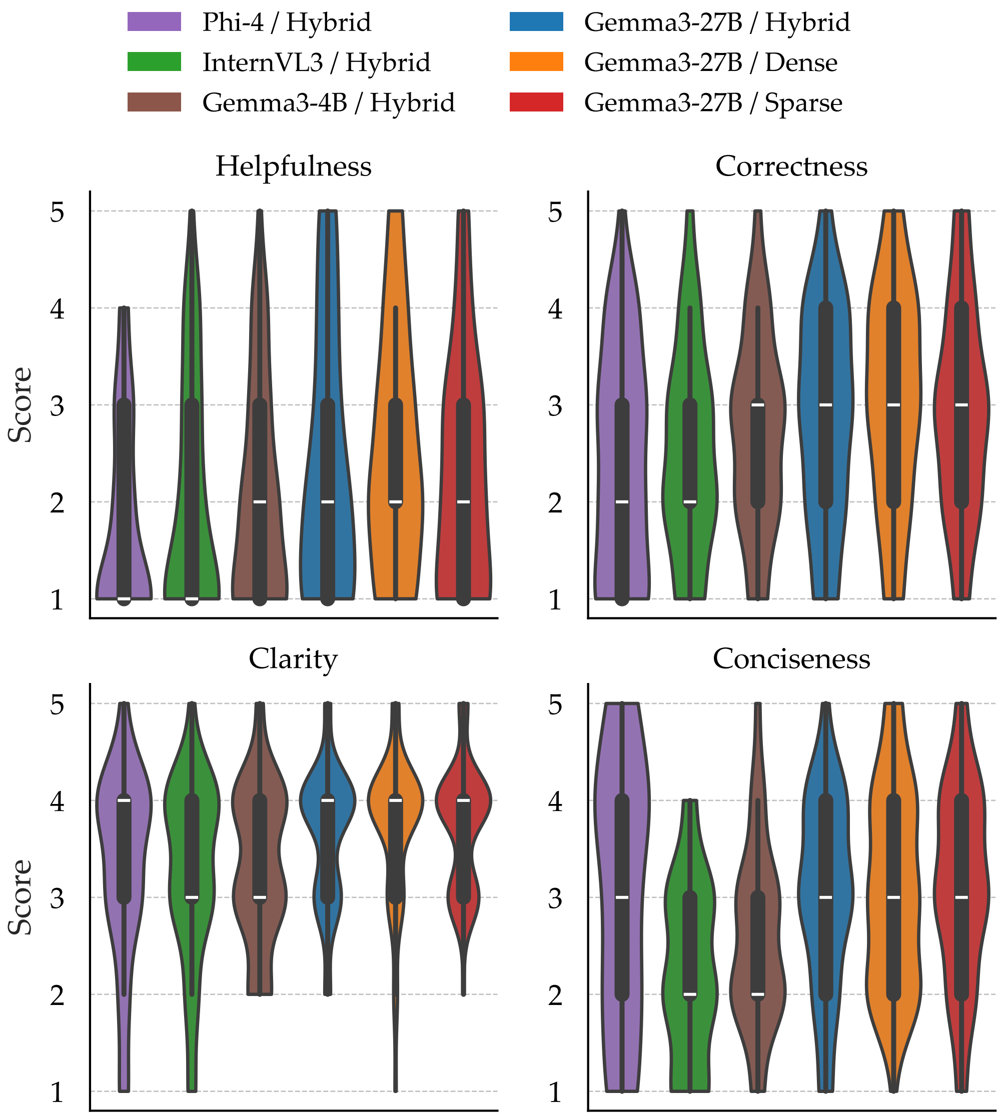


Figure 10: Distribution of human annotator scores per RAG configuration and annotation category, displayed using a combination of violin and box plots. The violin plot width indicates the density of scores, and the box plot shows the quartiles with the median marked in white.

If not denoted otherwise, the scores shown represent the median values of the three annotations. For better readability, VLM names are used in abbreviated form where no ambiguity occurs.

6.1. Overall Results

The overall score distributions for all six RAG setups across the four annotation categories are presented in Figure 10. All models score quite poorly on helpfulness with third quartile at three and median scores of one and two. Regarding correctness, the distributions are roughly symmetrical around the median values of three for the Gemma-3 setups, while Phi-4's and InternVL3's scores are skewed towards the lower end of the scale. Furthermore, the charts show medium to high scores on clarity. Looking at conciseness, Phi-4 slightly outperforms the Gemma3-27B setups, with lower scores for Gemma3-4B and particularly InternVL3.

6.2. Inferential Statistics

According to the research questions, the comparison of three subgroups of setups are of particular interest, i.e. (i) the three Gemma3-27B variants with different retrieval strategies, (ii) the models Gemma3-4B, Phi-4 and InternVL3 with consistent retrieval strategy, and (iii) Gemma3-4B and Gemma3-27B both with hybrid retrieval strategy.

6.2.1. Varying Retrieval Strategy

To test if at least one of the retrieval strategies show a significantly different distribution compared to the others, the Friedman two-way analysis of variance is applied on all four annotation categories (see Table 4). The hypothesis are compared against the significance level of $\alpha = 0.05$, which is a widely accepted standard in literature.

Hypothesis H_0: The distributions regarding the three different retrievers are the same within category: **Hypothesis H_a**: At least one distribution differs significantly from the others in category:	**p-value**	**reject H_0 in favor of H_a?**
Clarity	7.49e-1	no
Conciseness	6.38e-1	no
Correctness	2.67e-1	no
Helpfulness	3.02e-1	no

Table 4: Friedman two-way analysis of variance for varying retriever (i.e. Gemma3-27B / Hybrid, Dense and Sparse) compared to significance level $\alpha = 5.00\text{e-}2$.

As the table shows, all null hypothesis must be accepted and therefore no significant differences can be observed between the three setups.

6.2.2. Varying Generator Models

When comparing the generator models, the Friedman analysis hints at a significant difference on category *conciseness* (see Table 5).

Hypothesis H_0: The distributions regarding the three different generator models are the same within category: **Hypothesis H_a**: At least one distribution differs significantly from the others in category:	**p-value**	**reject H_0 in favor of H_a?**
Clarity	9.84e-1	no
Conciseness	2.28e-4	yes
Correctness	2.12e-1	no
Helpfulness	1.07e-1	no

Table 5: Friedman two-way analysis of variance for varying the generator model (i.e. Gemma3-4B, Phi-4 and InternVL3) compared to significance level $\alpha = 5.00\text{e-}2$.

To find out between which setups the differences in conciseness are significant, consecutive Wilcoxon matched-pairs signed-rank tests are conducted (see Table 6). Due to multiple tests on the same dataset, the probability of a type I error increases. Therefore, the Holm-Bonferroni compensation is used to reduce the significance level $alpha$ accordingly.

Index	**Hypothesis H_0**: The distributions does not differ within category *conciseness* between models: **Hypothesis H_a**: The distribution differs within category *conciseness* between models:	**p-value**	$\frac{\alpha}{n-i}$	**reject H_0 in favor of H_a?**	**effect size r_{rb}**
0	Phi-4 vs. InternVL3	1.78e-5	1.67e-2	yes	6.88e-1
1	Phi-4 vs. Gemma3-4B	7.65e-3	2.50e-2	yes	4.20e-1
2	InternVL3 vs. Gemma3-4B	3.09e-2	5.00e-2	yes	-3.68e-1

Table 6: Wilcoxon matched-pairs signed-rank test with Holm-Bonferroni correction on category *conciseness* for varying generator models. Significance level $\alpha = 5.00\text{e-}2$ and $n = 3$ tests.

All null hypothesis can be rejected, meaning the tests show significant differences between all models. The effect size favors Gemma3-4B over InternVL3, which are both exceeded by Phi-4.

6.2.3. Varying Model Size and Quantization

Comparing the Gemma3-4B versus its quantized sister-model with larger parameter size Gemm3-27B, a significant difference can be observed on categories *conciseness*, *clarity*, and *helpfulness*, as all null hypothesis are rejected (see Table 7). The difference in correctness on the other hand is statistically not significant.

Index	**Hypothesis H_0**: The distributions does not differ within the category between models: **Hypothesis H_a**: The distribution differs within the category between models:	**p-value**	$\frac{\alpha}{n-i}$	**reject H_0 in favor of H_a?**	**effect size r_{rb}**
0	Conciseness	3.86e-3	1.25e-2	yes	-4.77e-1
1	Clarity	8.23e-3	1.67e-2	yes	-4.68e-1
2	Helpfulness	9.28e-3	2.50e-2	yes	-4.73e-1
3	Correctness	1.71e-01	5.00e-2	no	

Table 7: Wilcoxon matched-pairs signed-rank test with Holm-Bonferroni correction on generator models Gemma3-4B and Gemma3-27B. Significance level $\alpha = 5.00\text{e-}2$ and $n = 4$ tests.

6.3. Annotator Agreement

To examine the overall quality of the annotation process, Krippendorff's Alpha is calculated for each RAG setup across all categories (see Figure 11). Almost all values are below 0.2, while some setups even show negative values on category *clarity*.

To further investigate the disagreement, scorings of three annotators are compared over their assigned sample items on category *clarity* (see Figure 12 for a representative example). Annotator 1 (red) mostly uses score 4 with some deviations, mostly in lower regions. Annotator 2 (green) scores generally lower, often even with inverse tendency (e.g. around samples 16 and 48). Annotator 3 (purple) mostly uses moderate scores between 2 and 4 (e.g. around samples 15 and between samples 40 and 50) with occasional use of 1 and 5.

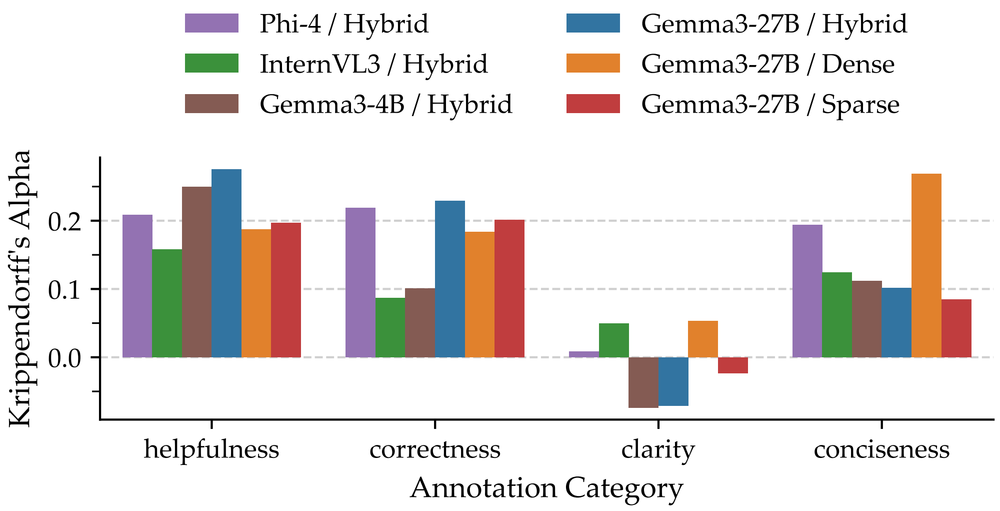


Figure 11: Krippendorff's Alpha for all RAG configurations on all annotation categories.

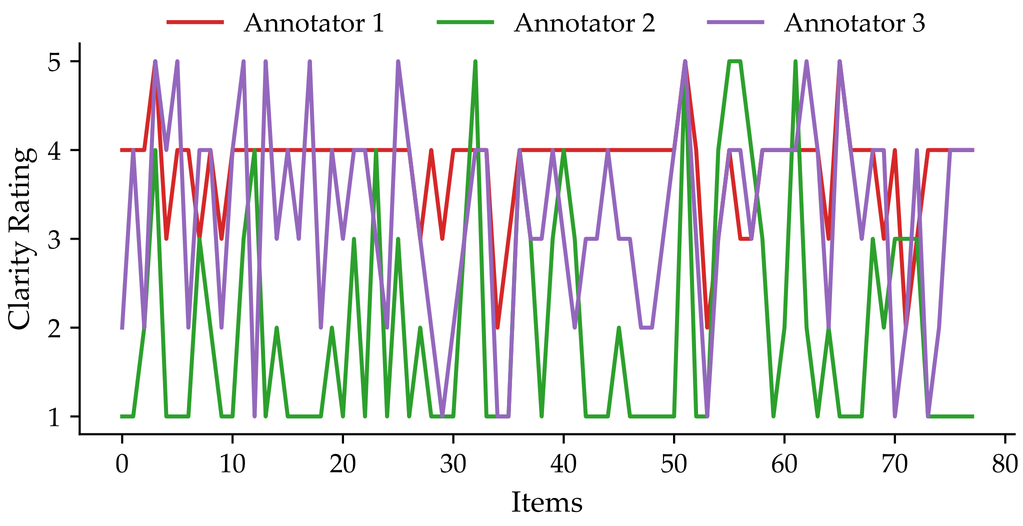


Figure 12: Clarity ratings of an annotator subgroup across all RAG configurations (13 questions times 6 setups makes up for 78 items in total).

7. Discussion

The study's results are discussed and contextualized in this section. The first three subsections cover retrieval, generation, and quantization of the generator, which corresponds directly to the research questions. Subsequently, overall performance comparisons are drawn and lastly, the limitations of this study are discussed.

Because this section includes detailed qualitative analyses, identifiers of the discussed samples are provided to ensure maximum transparency for readers wanting to dive deeper. To improve readability, the original request IDs are replaced by a new consecutive numbering system. The mapping table to the original request ID is provided in Appendix A.5.

7.1. Varying Retrieval Strategy

The statistics show no significant differences when replacing the retrieval strategy between dense, sparse, and hybrid. Using the Friedman two-way analysis of variance, none of the null hypotheses regarding the four annotation categories can be rejected (see Table 4). This result matches the distribution visualizations in Figure 10, with equal quantiles for all three setups on all categories with only one exception, which is the deviation of the dense retriever's first quantile in category helpfulness.

7.1.1. Initial Retrieval through Sparse and Dense Retrieval

For both the dense and sparse retrieval strategies, specific examples can be found, where they excel over their counterparts. To identify those cases systematically, the dataset was scanned for cases with highly diverging helpfulness scores for the different retrieval strategies. The retrieved chunks were then manually reviewed to assess their relevance and determine whether they led to helpful answers.

Qualitative Analysis When a user asked how to convert arrays to a proprietary datatype of a look-up-table block (request 1), the dense retriever was able to identify a function intended for this use case, which resulted in a helpful answer. The sparse retriever on the other hand retrieved chunks describing the look-up-table and general block usage, leading to an inaccurate answer.

In a contrary example, a user asked how to automatically calculate a checksum for CAN frames of the J1939[13] protocol (request 2). Using the sparse technique, the retriever was able to capture a sequence from the manual describing the full procedure of using a callback function and how to register it. The dense retriever on the other hand found a relevant function which suggested a checksum function for J1939, but missed the crucial chunk which explained how to use the callback.

[13]Serial Control and Communications Heavy-Duty Vehicle Network, doi: 10.4271/J1939_202306

Instead, several other chunks regarding checksum calculation of a different protocol (SENT[14]) were found, which introduced noise to the final answer.

In customer request 3, a very similar question was raised, but phrased less directly with much more noise and additional pictures. Excerpts of both requests are quoted here to illustrate their similarity in semantics and diverging syntax. For conciseness, the salutation and information regarding software versions are truncated:

Request 2:

> "[...] Is it possible to automatically generate the CRC value in a J1939 Canframe? [...]"

Request 3:

> "[...] Need to do a CRC calculation on the raw data of a CAN message. Do you have any support for this ? The message is defined with different datatypes inside and some space in between (Pic 1). In the auto code I got a data structure as in Pic 2. How do I reach the message data as an array of 8 bytes to use as input for the CRC calculation ? Is it possible through the TCanSigRecRoot pointer or... ? [...]"

Interestingly, only the dense retriever was able to identify the exact same, crucial chunk the sparse retriever was able to find in the example before. This produced the opposite result, where only the dense RAG system provided a helpful answer.

Challenges and Potential Improvements The requests 2 and 3 reveal a lack of robustness for both retrieval approaches and how sensitive they are to the exact phrasing of the request. This finding is unsurprising for the sparse BM25 algorithm, which relies on exact lexical matches. While this fundamental issue cannot be addressed directly, tuning the BM25's hyperparameters might increase precision as the saturation of term frequency and the penalty for long document length might need adjustment for this particular dataset (see Section 2.2.2). The design choice of using a stemming and cleaning algorithm prior to calculating the BM25 scores could also have been a cause for unreliability, as the uniqueness of domain keywords would be reduced. A simple test of a set of domain-specific words, abbreviations, and lines of code refuted this hypothesis though, as all terms remained unequivocal after applying the algorithm (see Appendix A.4). As the BM25 algorithm is reported to overly penalize long documents [LZ11], this was further investigated. A comparison between average chunk length of the sparse retriever (1048 words) and dense retriever (417 words) showed that this issue is not present here.

Regarding the dense retriever, the question arises whether the encoder is able to produce meaningful embeddings in the unknown domain, as it might deviate significantly from the pretraining data. An example from literature shows a dense retriever failing on the question "Who plays Thoros of Myr in Game of Thrones", as it

[14]SENT – Single Edge Nibble Transmission for Automotive Applications, doi: 10.4271/J2716_201001

cannot understand the critical semantics [Kar+20]. The requests 2 and 3 might hint at a similar issue, as the similarity of questions seems obvious for a domain expert and yet the dense algorithm fails in one of the cases. One countermeasure proposed in the literature to address this challenge directly is fine-tuning the encoder on the unknown domain [Gao+24]. Beyond domain-specific adaptation, several general measures exist to enhance retrieval quality, applicable to both sparse and dense approaches.

One such measure is to instruct the generator to decompose the initial retrieval query into more distinct subqueries [Shi+24]. Following this approach, noisy or convoluted questions can be broken down into clear pieces with the goal of increasing retrieval quality. However, as the generator is generally not trained on the proprietary domain, its ability to rephrase precisely requires further investigation. Another enhancement to the RAG pipeline involves multiple retrieval steps, where an abort criterion is checked after the usual steps of retrieval and generation to determine whether a subsequent run is needed. Iterative retrieval augments the query with previously generated answers, whereas recursive retrieval decomposes the query using chain-of-thought methods (see Section 2.5.3) to increase information specificity [Gao+24].

Quantifying Retrieval Failure To illustrate the magnitude of the retrieval failure issue, a quantitative analysis is applied: 30 of the 65 samples in the dataset are annotated with a helpfulness score of 1 or 2. Even though the initial retrieval might not be the single cause of failure, as revealed below, it highlights failed initial retrieval as a key weakness. By further investigating the requests with helpfulness scores of 1, the points of failure could be further narrowed down to (i) the retriever misses the similarity between query and relevant chunk or (ii) the crucial piece of information is missing in the database (see Appendix A.7). While the less frequent case (ii) obviously suggests completing the database with the respective documentation, the more dominant case (i) stresses the need for measures discussed above.

7.1.2. Hybrid Retrieval and Fusion Strategy

Besides the issue of initial retrieval failure, the discussed requests 2 and 3 show that sparse and dense retrieval excel and struggle with different queries. This strengthens the argument to exploit both strategies in a hybrid approach, which stands in accordance with the literature attesting an increase in robustness [Gao+24] [MM24]. To investigate this claim, selected requests are listed and analyzed in Table 8. Besides the already discussed precision issues of the sparse and dense retrievers, additional points of failure are revealed. A general observation across all strategies is the generator occasionally ignoring the relevant retrieved chunk, which obviously plummets the answer's quality (requests 1, 6, 7). In the case of request 4 this issue even occurred, when the user exposed the crucial piece of information themselves, while all retrievers provided irrelevant context. Therefore, the generator model struggles

ID	Sparse	Dense	Hybrid	Comment
1	- / ×	2 / ✓	3 / ×	Generator of hybrid-RAG ignores relevant chunk.
2	2 / ✓	- / ×	- / ×	Relevant chunk gets sorted out with hybrid strategy.
3	- / ×	1 / ✓	1 / ✓	The fusion strategy of hybrid retrieval correctly reorders the relevant chunk.
4	- / ✓	- / ×	- / ×	In the sparse setup, the answer was extracted from the information the user provided. In the other cases, irrelevant chunks were reproduced.
5	1 / ✓	3 / ✓	2 / ✓	The fusion strategy of hybrid retrieval correctly reorders the relevant chunk.
6	- / ×	1 / ×	1 / ✓	Generator of dense-RAG ignores the relevant chunk. The relevant chunk is correctly reranked in hybrid fusion strategy.
7	2 / ×	- / ×	- / ×	Generator of sparse-RAG ignores the relevant chunk. Relevant chunk gets sorted out with hybrid strategy.

Table 8: Ranking of relevant chunk with different retrieval strategies for selected example requests; columns Sparse, Dense and Hybrid show rank of relevant chunk if retrieved, else -, and if relevant chunk was used in the answer: ✓, ×.

with processing the given information, which is further discussed in the next section. In other examples, the reranking strategy failed when using the sparse and dense approaches. Requests 2, 6, and 7 show that even though the crucial chunk was retrieved initially, it was discarded in the hybrid retriever's fusion step. With the current hybrid approach, a helpfulness score of 4 or 5 could be achieved on 15 out of 65 requests, while in 21 cases, at least one of the three retrieval approaches could achieve the same scores. This strongly suggests that the simple fusion strategy based on a weighted sum is insufficient and should instead be replaced by a more advanced post-retrieval component, as proposed in the literature [Gao+24]. These specialized reranking models process the query and chunk candidates side-by-side and are able to capture nuances and details, which the initial retrieval step might miss.

7.1.3. Summary

While the statistical differences between the retrieval strategies are insignificant, the qualitative analysis reveals several issues regarding each individual approach. What intuitively seems like a contradiction can be explained as follows: The sparse and dense retrievers, with their particular strengths and weaknesses, excel on different queries, as illustrated with the requests 2 and 3 above. Over the whole dataset, however, these differences balance each other out, resulting in statistical insignificance. As the fusion step of hybrid retrieval suffers from its own challenges, it also cannot fully exploit the potential of combining both strategies, leading to assessment scores ranging in the same quantiles. In addition, the overall weak performance combined with the low sample size of 65 requests might disguise further subtle discrepancies.

7.2. Varying Generator Models

Varying the generator of the RAG pipeline between Gemma3-4B, InternVL3, and Phi-4 shows only statistically significant differences on the category conciseness (see Table 5). Further analysis reveals Phi-4 surpassing Gemma3-4B, which is in turn more concise than InternVL3 (see Table 6), with medium effect sizes (0.42, 0.37). The effect size between Phi-4 and InternVL3 can be classified as large (0.69), which matches the clear distribution differences in Figure 10, with higher second and third quartiles as well as overall favorable distribution of Phi-4.

7.2.1. Qualitative Analysis

To further analyze this finding qualitatively, InternVL3's answer to request 4 is investigated showing examples of issues leading to low conciseness scores. The model advises the user to consult the documentation or to contact the support, which was assessed as noise, because it presents no valuable information. In addition, it recommends to use a specific tool for testing[15], which is not particularly wrong, but completely unspecific and overly generic. Another pattern observed in other answers was the repetition or summary of the request, which again brings no value.

While the exact reason for Phi's more concise answers cannot be explained with certainty here, one possible factor is the model size. Phi-4 has the smallest VRAM footprint of 11.9GB among the models, compared to approximately 15GB for InternVL3 and 18.5GB for Gemma3-4B, leading to speculation about a correlation with the models' verbosity. Additionally, Phi's technical report emphasizes the high quality datasets used for training, which might also reduce noisy answers [Mic+25]. In general, Microsoft seems to aim for compactness and efficiency regarding the Phi-4 family with maximum parameters over all models at 14B [Abd+24]. Gemma3-4B and InternVL3 on the other hand are part of model families with larger sister models up to 27B and 78B [Tea+25] [Zhu+25], so the overall development emphasis might be less focused on efficiency. If, on the other hand, this is set as central development goal, there is value in keeping the answers short, to lower latency and memory footprint. The latter is directly linked to the answer length, because the kv-cache grows with each token generated.

Beyond conciseness, the models showed remarkably similar performance across the other categories helpfulness, correctness, and clarity. Even the qualitative anal-

[15]The tool's name is anonymized.

ysis could reveal systematic differences. This observation stands in accordance with the benchmarks used for selecting the models 11, where all chosen candidates score in comparable ranges. Additionally, the previous section showed that the retrieval was often insufficiently precise, which might disguise some model differences, as it often led to dissatisfying answers of all models. This suggests rerunning the experiment on a different set of questions.

7.2.2. Unfamiliar Domain and Context Misjudgment

Deeper qualitative analysis shows that the generator models were not reliably able to assess the relevancy of retrieved chunks, which is demonstrated in the following example. Answering customer request 8 regarding a PWM frequency issue, Gemma3-4B integrated a passage about flashing a controller with a debugging device into its answer, which stands in no relation to the question whatsoever. Moreover, the prompt explicitly instructed the models to decline answering, if the provided context was irrelevant, but only three times overall, the models identified those cases (request 7: Gemma3-27B, request 9: Phi-4, request 10: Gemma3-4B). This stands in contrast to the finding of poor retrieval precision of the previous section, which implies that the models should have chosen this option more often. Therefore, the models either misjudge irrelevant information or ignore the prompt instructions. Another observed issue is models ignoring the relevant chunk, even though it was part of the given context, which was already mentioned in the previous section (see Table 8: requests 1, 4, 6, and 7).

The challenge of generator models ignoring relevant information was already observed in related publications. One possible issue might be the lost in the middle problem, which states that LLMs tend to focus heavily on the beginning and end of provided context, while neglecting the middle part [Liu+23]. As the chunk-by-title strategy potentially creates long sequences, this could very well be a root of the stated problem. In addition, the multimodality of this thesis' approach could mislead the focus even further, even though this is speculative as no literature covering this hypothesis could be found.

However, the qualitative analysis of request 8 heavily implies that another issue is the generators struggle with the unfamiliar domain. Poor assessment of relevance and weak deduction of relations cause the models to fail to integrate the crucial information or to add unnecessary and unrelated noise into their answers. Remedy for this problem could be (i) training the models on the proprietary data, (ii) increase relevancy of retrieved content, and (iii) reduce the retrieved content to a minimum. While (i) stands in fundamental contrast to the decision to use a RAG approach in the first place, (ii) and (iii) at least partly address the issue of the models' wrong focus by detaching the judgment of relevancy from the generator to an upstream component. This component could be implemented by following the measures of the previous section, i.e. using query rewriting techniques for improved retrieval and cross-encoding for more sophisticated reordering and fusion. Under the assumption that the relevancy can be increased by these measures, the reduction of overall context (iii) can be simply reached by only using the highest ranked chunk, instead of best three as of now. Another option, which needs further investigation, is to reduce the chunk length by splitting current chunks into smaller units or switching the entire chunking strategy. An additional suggestion in the literature is context compression, a technique to detect and remove unimportant tokens, which does not affect the generator's understanding [Gao+24].

As none of the models can outperform their counterparts overall and all show similar points of failures, Phi-4 might be the most suitable candidate, if conciseness is crucial. In order to explore more subtle differences, rerunning the survey on a different set of questions might be expedient.

7.2.3. Summary

The tested generator models showed generally similar performance. However, Phi-4 answered statistically significantly more concisely than its competitors, which might correlate with its notably small VRAM footprint and development focus on efficiency and high-quality training data. Reviewing model answers suggested that lower conciseness scores resulted from repetitions or summaries of customer request as well as overly generic advice.

Overall, all models struggled with the unfamiliar domain. The qualitative analysis revealed that models completely misjudged the relevance of provided information, as they were observed to integrate unrelated content into answers. Additionally, they occasionally ignored the crucial chunk, even when it was successfully retrieved.

To address these issues, the methods proposed in Section 7.1 should be applied, specifically utilizing query rewriting and cross-encoding to increase the relevance of retrieved information. Furthermore, context compression techniques could further enhance performance, which remove unimportant tokens, while preserving the model's understanding.

7.3. Varying Model Size and Quantization

The comparison between Gemma3-4B and the quantized Gemma3-27B shows significant differences on three categories: Gemma3-27B exceeds its smaller counterpart regarding conciseness, clarity, and helpfulness. With absolute effect sizes of approximately 0.47 for each category (see Table 7) and reference values from literature, these effects can be classified as 'medium' [IL24].

These findings are also confirmed in Figure 10 showing skewed distributions of Gemma3-27B (blue) towards the upper end of scale for the mentioned categories compared to Gemma3-4B (brown). The difference in helpfulness is only slightly superior, with matching quantiles for both models, but the distribution shows a little thicker concentration towards scores of 4 and 5. Regarding conciseness, the distributions noticeably tend towards opposite ends of the scale with higher second and third quantile by one point for the quantized model. Even though the clarity medians differ as well, this finding should be discarded due to insufficient reliability originating from systematic annotator disagreement. This can be concluded from the negative values of the Krippendorff's Alphas regarding clarity for both models (see Figure 11). For further discussion of annotator agreement see Section 7.5.

7.3.1. Qualitative Analysis

The overall better performance of Gemma3-27B can also be observed when looking at a specific example question (request 11). The customer asked the RAG system to clarify two configuration options regarding error detection. The relevant information was retrieved in a chunk containing two diagrams (see Figure 13). Gemma3-27B combined all required information about 'error', 'set condition' and 'detection method' to one sophisticated answer, while Gemma3-4B completely ignored the information regarding 'detection method' resulting not only in an imprecise but wrong answer.

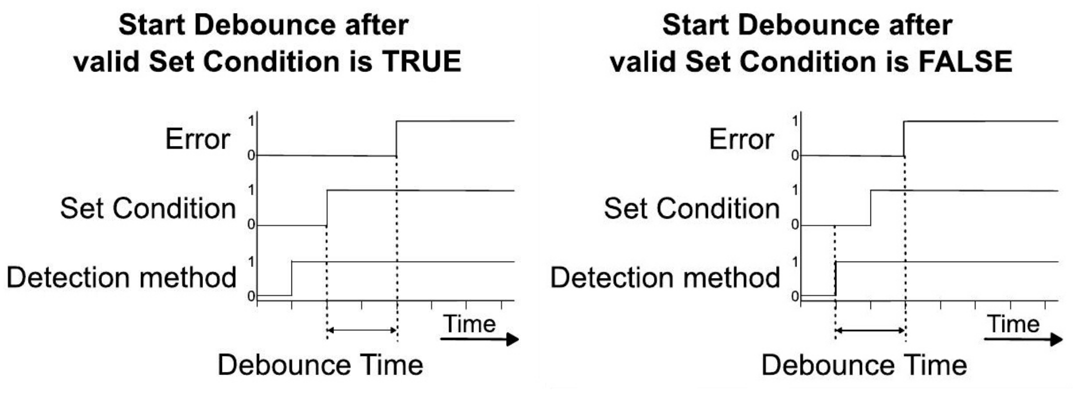

Start Debounce after valid Set Condition is TRUE
Start Debounce after valid Set Condition is FALSE
Error
Set Condition
Detection method
1
0
Time
Debounce Time

Figure 13: Chunk containing a graph, which was interpreted correctly by Gemma3-27B, whereas Gemma3-4B ignored the information regarding 'detection method'.

7.3.2. Contextualizing the Finding

Overall, not only the statistical data but also a qualitative example indicates that the larger, quantized model outperforms its smaller sister model. This finding is coherent with other studies from literature. Dettmers and Zettlemoyer investigated the correlation between model size, quantization and zero-shot accuracy, i.e. the accuracy on tasks where no examples are provided and the model needs to use its learned knowledge. The study was a comparison of quantized models with higher number of parameters and their counterparts with lower number of higher-precision parameters. Under the condition of comparing models variants of equal size (total model bits) and under the constraint of parameters being represented by at least four bits, the models with larger number of quantized parameters outperformed their counterparts in almost all cases. In the case of sub four bit quantization, a sharp drop-off in performance could be observed for two out of four models with large numbers of parameters [DZ23].

Another study examined different quantization techniques on the Qwen model family. Varying the number of parameters and quantization on different benchmark tests to evaluate multiple abilities, the authors come to the same conclusion as the other comparative study: While quantized models with larger parameter size are preferable in general, significant performance decrease can be observed in some cases for quantization of three bits or below [Jin+24].

However, there are two noticeable differences regarding the cited study designs and the present thesis' study, which actually strengthen the argument that in most cases the quantized models outperform their sister models of similar total size. The first difference is that a model in a RAG setup is enhanced with additional information or examples, whereas the zero-shot accuracy is the opposite case, where no additional information is given. For both cases, the literature and the present thesis implicate that quantized models can outperform their counterparts of similar total size. The second difference in the studies lies in the quantization strategy, which is more advanced in the present thesis. The GGUF mixed-precision block-wise strategy generally quantizes the weights down to two bits, but preserves higher precision by storing scaling factors and offsets for smaller block-sizes, compared to storing them for the whole layer. Additionally, more critical weights are identified and quantized less aggressively, which again preserves precision.

The observed high performance in conjunction with this advanced quantization strategy suggests that the four-bit threshold for minimum parameter precision might not be universally applicable. For more complex quantization methods, it might be necessary to develop more advanced measures to characterize the minimum required precision. These findings confirm that (i) larger quantized models outperform smaller counterparts on different tasks and (ii) that advanced quantization strategies offer further potential in pushing boundaries.

7.3.3. Summary

The quantized model with higher parameter count – Gemma3-27B – outperformed its counterpart of equal size (total model bits) – Gemma3-4B. The superior performance is reflected in significantly increased helpfulness and conciseness scores with effect sizes of classification 'medium', and also in a qualitative analysis, where only the quantized model was able to interpret a graph correctly. The difference in clarity was also significant, but not further considered as the annotator agreement for this category was particularly low.

This finding is consistent with other use cases in the literature [DZ23]. Moreover, a comparison to the literature suggested different quantization limitations depending on the exact technique. While the literature claimed that at least 4-bit precision should be preserved, the model in this thesis was quantized with a more sophisticated strategy, which compresses down to 2-bit and still showing strong performance.

7.4. Contextualizing the Overall Performance

The last three sections of the discussion were closely tied to the research questions, focusing on the retriever, generator and quantization of the latter. This section on the other hand abstracts from specific RAG components, putting the overall performance into perspective to a plug-and-play solution in GPT4All[16] and to a set of synthetic questions. Subsequently, a comparison with a published study assessing a similar approach on a comparable domain is conducted.

Comparison to a Plug-And-Play Solution GPT4All is an open-source solution desktop application, which allows running a RAG pipeline locally. Even though only a text-based approach without vision capabilities is supported, it can serve as a benchmark of an easy to setup, out-of-the-box solution.

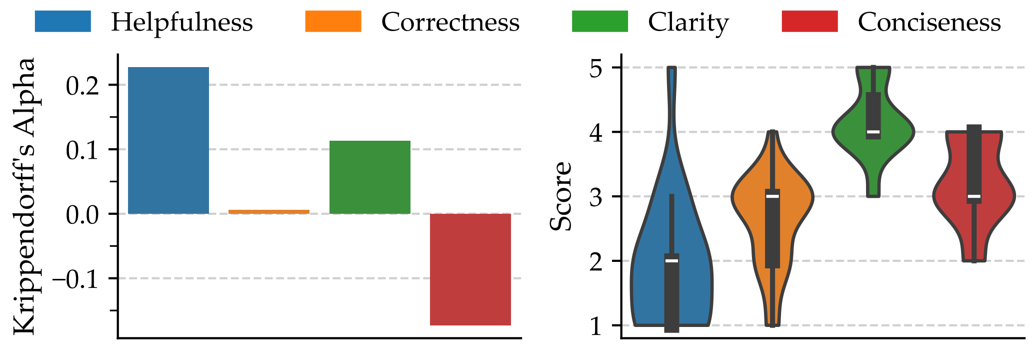


Figure 14: Krippendorff's Alpha and score distribution for LLama-3.1-8B-Instruct-q4_0 running with GPT4All on a subset of the original questions.

For this investigation, all configurations are kept to default, with retrieving five chunks of maximum 512 token length using nomic-embed-text-v1.5, while LLama-3.1-8B-Instruct-q4_0 serves for generation. The setup is assessed on a subset of 23 questions – due to limited availability of annotators for this additional annotation process – but otherwise with the same methodology as before. The helpfulness distribution shows an unsatisfactory performance in almost all cases with third quantile at score of 2 (see right side of Figure 14). As the exact reasons for the performance difference between this basic approach and this thesis' pipeline – integrating vision, a hybrid retrieval strategy, and hierarchical chunking – are not the subject of this work, the gap suggests future research in this area.

[16]https://www.nomic.ai/gpt4all

Comparison to Synthetic Questions – Direct and Without Noise For an additional investigation, a set of 30 synthetic questions were directly derived from the database's documents. Each question was clearly formulated with domain specific terms and without noise. The answers of the Gemma3-27B setup with hybrid retrieval were assessed once again showing median performance scores of 4 or better over all categories (see right side of Figure 15). The significantly better performance on the well stated synthetic questions compared to the initial noisy and convoluted questions demonstrates the impact of question formulation on answer quality. This strengthens the argument for further investigating question rewriting techniques to decompose the query into clear subqueries (see Section 7.1).

The annotator agreement for both these additional experiments – GPT4All and the synthetic dataset – is analyzed alongside the main results in the next section.

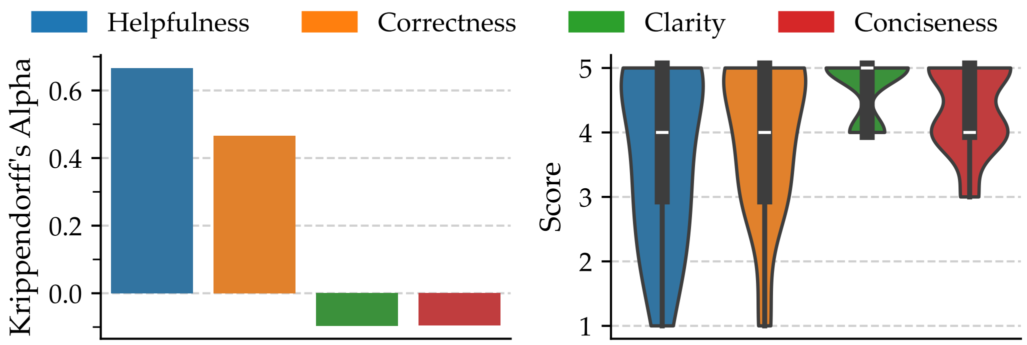


Figure 15: Krippendorff's Alpha and score distribution for Gemma3-27B / Hybrid on the synthetic question dataset.

Comparison to the Literature The rest of this section draws a comparison to a publication which evaluates a RAG pipeline on a dataset of software manuals and documentation of programmable controllers, making it a comparable domain to the one subject of this thesis [RL24]. The publication assesses different pipeline configurations, whereas for the comparison to this thesis, only the best performing is considered. It follows a text-image approach, where images are summarized into text using a multimodal generator model, before text and image summaries are embedded into a shared database. OpenAI's GPT-4V and textembedding-3-small are chosen for generation and text encoding. 100 test questions were synthesized from the documents selected as database.

The answers were assessed on a binary scale and a final score for the whole set was determined by averaging. The results show a score of 0.43 in correctness and 0.86 in relevancy compared to 0.18 and 0.96, when using the generator alone without RAG. Therefore, adding RAG resulted in improvements in the former category by 25 percent points and slightly decreased performance on the latter [RL24]. As the category definitions differ from the one used in this thesis, a direct side-by-side comparison is inappropriate. However, for a rough, qualitative comparison, a helpfulness score of 4 or 5 could be considered a satisfying answer in this thesis' study. Gemma3-27B with hybrid retrieval could generate such an answer in 15 out of 65 cases (23.1%) on the initial dataset. Furthermore, in 21 cases, at least one of the retrieval strategies could find the crucial chunk which shows even more unexploited potential (see Section 7.1). Even though the models were not evaluated without RAG in this thesis, it seems appropriate to assume they would fail entirely due to the proprietary and therefore completely unfamiliar domain.

Despite the differences in annotation categories and datasets, the comparison between this thesis' RAG approach and the one proposed in the publication indicates that none of the two significantly outperforms the other. In both cases, the improvement achieved using RAG compared to just a standalone generator model can be estimated to roughly 25 percent points. Therefore, this thesis' pipeline demonstrates competitive performance with an approach utilizing GPT-4V, which is by a factor 75 larger than the largest model used in this thesis[17]. This supports the hypothesis that sizing up the generator model has limited effects as it cannot compensate for poor retrieval quality. In addition, the examples provided in the publication indicate that the synthesized questions are clearly and directly stated without any noise, whereas in this thesis real customer questions are used for evaluation which contain such noise, making them harder to process.

7.5. Limitations

This thesis' study has limitations that affect the generalizability and interpretation of the discussed findings.

Sample Size The main study is based on 65 questions taken from real customer requests. While each question itself is therefore of high validity, the overall sample size is fairly low. The limited sample size was a result of the decision to conduct human evaluation within resource constrains. This weakens statistical statements considerably and precludes drawing strong generalized conclusions from the findings. In addition, the severe performance difference on the synthetic dataset revealed a strong correlation between question and answer quality, which further prohibits generalization across question types. However, the real customer requests can serve as a realistic upper threshold on the domain, both with regards to content and expression complexity.

Annotator Agreement Another noticeable issue is the overall low inter annotator agreement. In the main study, the Krippendorff's Alphas revealed strong disagreement across all annotation categories and particularly regarding clarity (see Figure

[17]While the exact size is not published, different sources estimate it to 1.8T parameters, which would make it larger by factor 75 compared to Gemma3-27B, which is the largest model used in this thesis.

11). Compared with reference values, the data must be seen as unreliable, as all Alphas are significantly lower than the threshold of 0.67. This argument is strengthened by looking at the clarity ratings of a subgroup of annotators across all RAG setups (see Figure 12), which shows that annotators disagreed on almost all items.

This result strongly suggests that annotators did not have a common understanding of the annotation categories and the influence of personal interpretation was predominant. Despite implementing standard measures to increase clarity among the annotators (briefing, provision of examples, easy to access annotation guidelines), the observed agreement was low. The author of this thesis acknowledges though that definitions of some annotation category values leave room for interpretation, which in combination with the challenging task of generated free text assessment is insufficient. For future studies, these definitions need refinement to explicitly distinguish each value with certainty or supported by decision trees to guide annotators.

Besides the deficits in study design, it should be noted that Krippendorff's Alpha is reported to be very sensitive to skewed data [QL16]. This issue can be observed for the clarity rating of the follow-up study on the synthetic dataset (see Figure 15). While the clarity distribution is highly skewed with scores of 4 and 5 exclusively, the Krippendorff's Alpha shows a negative value indicating systematic annotator disagreement, which seems contradictory. While the exact reasons for Alpha's sensitivity are not discussed here, it shows that its interpretation without looking at the data distribution can lead to false conclusions. Further literature even calls for the development of better annotator agreement measures in general, because of several reported issues [Zha+18]. Overall, the indication of Krippendorff's Alpha of low annotator agreement seems plausible for this study as it corresponds with the overall impression gained from Figure 12.

If the ratings diverge as significantly as in the case of the main study's clarity scores, the data was not further considered (see previous subsections). For the remaining categories the data is only used for rough indications, which were all further investigated by an extensive qualitative analysis. The data therefore guided the direction, but was never used solely as evidence for any hypothesis. Therefore, the qualitative analysis is still valuable as it contributes to confirming general challenges encountered with RAG setups.

Further Limitations The RAG pipeline was run offline on constrained hardware resources (see Appendix A.2), which limited the size of the examined encoder and generator models. This might impact the overall performance, as larger models are generally more capable [Kap+20].

Furthermore, all models or algorithms used in this thesis are published under Apache[18] or MIT[19] license, or under Gemma Terms of Use[20]. Therefore, only models that are free-of-charge for both academic and commercial use were examined, which

[18] `https://www.apache.org/licenses/LICENSE-2.0`
[19] `https://mit-license.org/`
[20] `https://ai.google.dev/gemma/terms`

excludes promising proprietary solutions.

Another limitation is the documentation which builds the database for the retrieval step. It cannot be guaranteed that all information is comprehensively explained and that no gaps exist. Additionally, the documentation is not optimized to be processed by language models. It is therefore possible that some performance issues are caused by the overall documentation quality.

7.6. Points of Failure and Potential Improvements

This section sums up the five key points of failure identified in the discussion (marked red in Figure 16), alongside three additional enhancement options from literature (marked green).

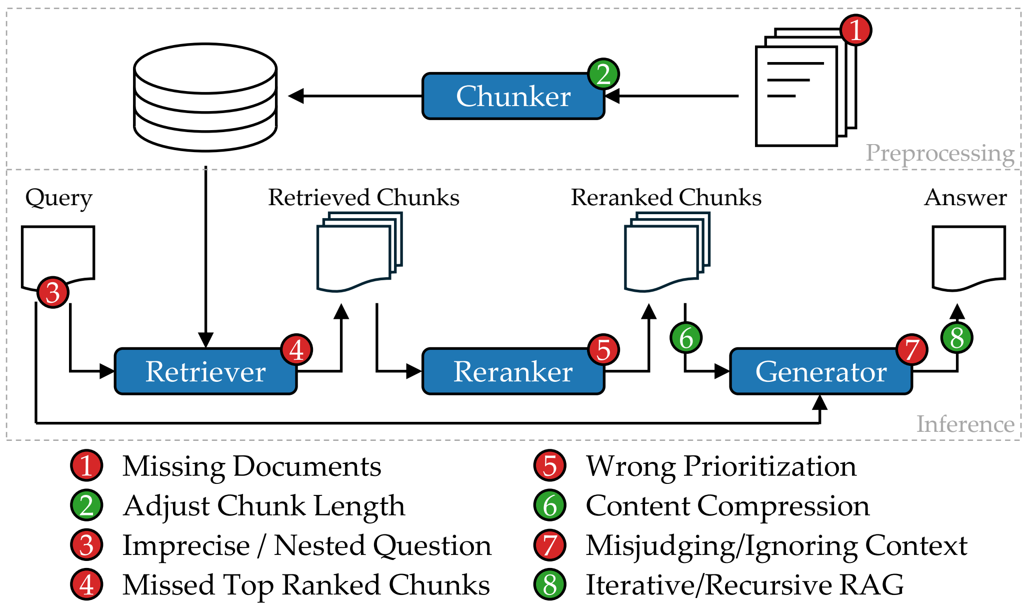


Figure 16: RAG pipeline with highlighted points of failure identified in the discussion (red), which in themselves are places for improvement. Additional enhancements suggested in the literature are marked in green. Figure adapted from Barnett et al. [Bar+24].

The issue with probably the greatest influence on answer quality is insufficient initial retrieval, which can have several causes. One simple reason of failure originates from missing information in the database (1) which lowers answer quality directly. Even though only two cases were explicitly observed in this study, addressing this issue is labor-intensive, because complementing the documentation must be done manually and proving the completeness is impossible. An additional variable is how the documents are chunked (2). In this thesis, semantic chunking was applied, where structural elements like titles, subtitles, and beginnings of headers in code determined the cut and therefore the chunk length. As the influence of the chunking strategy was not explicitly investigated for this thesis' use case, it is left open for future research. Another impact on retrieval quality is the query (3), as it might be convoluted, imprecise, and ambiguous. Query rewriting techniques can address this by breaking down the core matters while abstracting from noisy formulations. An additional performance comparison between the initial user requests and well-stated synthetic questions revealed major differences in favor of the latter, suggesting that query rewriting techniques offer significant potential for future exploration.

The final influence on chunk relevance is the retriever component itself (4). Both sparse and dense techniques showed success and failure on different queries, which confirmed combining them to be a promising approach. For the sparse retrieval the BM25 algorithm was utilized, which offers room for improvement by tuning its hyperparameters. The dense retrieval might have failed occasionally, because the encoder may not have been able to provide meaningful embeddings on the unknown domain. Examining this hypothesis is also a future task and if confirmed, would imply training the encoder on the proprietary data.

The reranking strategy in the hybrid retrieval approach was identified as another point of failure (5). For assessing the relevance of retrieved chunks from sparse and dense strategy, the similarity scores were normalized and averaged over both strategies. It was occasionally observed that crucial chunks were falsely prioritized low, which suggests refining this approach. A promising alternative for future investigation is the use of a cross-encoder, which judges the relevance by processing each chunk side by side with the initial query.

A further area for improvement is the generation phase (7). As observed in several examples, the generator ignored relevant context in favor of noise and occasionally integrated unrelated information into an answer. The latter issue strongly hints at the generator struggling with the unknown domain, which can only be explicitly addressed by training the model. Alternatively, the aforementioned measures to increase the relevancy of provided chunks might already provide a remedy, but need further investigation. Additionally, the content can be compressed (6) with a technique which eliminates less important words while maintaining semantics at the same time. To explore whether long context is then processed more effectively remains a subject for future work.

For particularly complex questions, advanced RAG concepts should be evaluated (8), where based on the previous generated answers, query refinements and additional retrieval steps are conducted to iteratively or recursively converge towards a more sophisticated answer.

8. Conclusion

This section concludes the thesis with a summary and recommendations for future research.

Summary This feasibility study investigated different multimodal RAG setups regarding their answer quality within limited computational resources on a proprietary domain. The RAG setups differed in retrieval strategy, generator models, and quantization of the latter. For evaluation, human annotators assessed the answers for a test set of anonymized customer requests.

The results showed that the sparse and dense retrieval strategies (BM25 and text-image-encoder) succeeded and failed on different queries. This suggests combining both strategies in a hybrid approach. Apart from that, both sparse and dense retrieving strategies failed to find the crucial chunk in several cases, even though it could be manually verified that it was existent in the dataset.

The tested generator models showed comparable performance, with Phi-4 answering more concisely than its competitors. Additionally, the quantized model with higher parameter count outperformed its counterpart of comparable total size. While the generator models were able to produce helpful answers, when appropriate chunks were retrieved, they occasionally misjudged or ignored the provided content.

The overall success rate of providing helpful answers was therefore generally low (23%, depending on exact setup), which could be traced back to the struggle with the unknown domain of both the retriever and generator, and to the request complexity. A test on a set of synthetic questions revealed a substantial increase in performance compared to the real customer inquiries (median helpfulness score of 4 vs. 2 on a 5-point scale), which highlighted the impact of noisy and ambiguous formulations in the queries.

Despite these challenges, running a RAG pipeline proved to be feasible on consumer hardware with free-of-charge models and libraries, making it accessible for small businesses.

Recommendations For Future Research Based on the RAG setup of this thesis, some design decisions proved effective, while several points of improvement have been identified that address the weaknesses of the current approach and point the direction of future research.

As noted above and confirmed by related literature [DZ23], larger quantized models should be preferred over counterparts given equivalent total size constraints.

While building on the strengths of hybrid retrieval can be generally recommended, the fusion of the sparse and dense chunks needs refinement and calls for a more sophisticated approach, e.g. by using a cross-encoder. To enhance the initial retrieval quality, query rewriting techniques could be a significant enhancement, as the performance on synthetic questions was substantially increased compared to the real customer queries. Future research should investigate whether models without additional domain knowledge are able to provide adequate rewriting. To answer complex inquiries, advanced RAG methods such as recursive retrieval may be beneficial. If these measures prove insufficient, the more resource-intensive approach of fine-tuning the retriever and generator needs investigation.

As this study was limited to consumer hardware and free-of-charge models, future research could examine cost-benefit tradeoffs when exploring commercial models.

Given the observed performance difference on synthetic and real-world inquiries, researchers are recommended to test on real-world questions whenever possible, as only those provide realistic complexity.

9. References

[Abd+24] Marah Abdin et al. *Phi-4 Technical Report*. 2024. arXiv: 2412.08905 [cs.CL].

[AC24] Bhashithe Abeysinghe and Ruhan Circi. *The Challenges of Evaluating LLM Applications: An Analysis of Automated, Human, and LLM-Based Approaches*. 2024. arXiv: 2406.03339 [cs.CL].

[AD18] Ergün Akgün and Metin Demir. "Modeling Course Achievements of Elementary Education Teacher Candidates with Artificial Neural Networks". In: *International Journal of Assessment Tools in Education* 5 (Jan. 2018). DOI: 10.21449/ijate.444073.

[Ama24] Xavier Amatriain. *Prompt Design and Engineering: Introduction and Advanced Methods*. 2024. arXiv: 2401.14423 [cs.SE].

[AT22] Eren Can Aybek and Cetin Toraman. "How many response categories are sufficient for Likert type scales? An empirical study based on the Item Response Theory". In: *International Journal of Assessment Tools in Education* 9.2 (2022), pp. 534–547. DOI: 10.21449/ijate.1132931.

[Bai+25] Shuai Bai et al. *Qwen2.5-VL Technical Report*. 2025. arXiv: 2502.13923 [cs.CV].

[Bar+24] Scott Barnett et al. "Seven Failure Points When Engineering a Retrieval Augmented Generation System". In: *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering - Software Engineering for AI*. Lisbon, Portugal: Association for Computing Machinery, 2024, pp. 194–199. ISBN: 9798400705915. DOI: 10.1145/3644815.3644945.

[BGI23] Sebastian Bruch, Siyu Gai, and Amir Ingber. "An Analysis of Fusion Functions for Hybrid Retrieval". In: *ACM Trans. Inf. Syst.* 42.1 (Aug. 2023). ISSN: 1046-8188. DOI: 10.1145/3596512.

[Bor+24] Florian Bordes et al. *An Introduction to Vision-Language Modeling*. 2024. arXiv: 2405.17247 [cs.LG].

[CCG21] Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. *Evaluation of Text Generation: A Survey*. 2021. arXiv: 2006.14799 [cs.CL].

[Che+22] Wenhu Chen et al. "MuRAG: Multimodal Retrieval-Augmented Generator for Open Question Answering over Images and Text". In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Ed. by Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 5558–5570. DOI: 10.18653/v1/2022.emnlp-main.375.

[Che25] Ruixi Chen. “Retrieval-Augmented Generation with Knowledge Graphs: A Survey”. In: *Submitted to Computer Science Undergradaute Conference 2025 @ XJTU*. 2025. URL: `https://openreview.net/forum?id=ZikTuGY28C`.

[Chk+24] Zina Chkirbene et al. “Large Language Models (LLM) in Industry: A Survey of Applications, Challenges, and Trends”. In: *2024 IEEE 21st International Conference on Smart Communities: Improving Quality of Life using AI, Robotics and IoT (HONET)*. 2024, pp. 229–234. DOI: `10.1109/HONET63146.2024.10822885`.

[Cob+21] Karl Cobbe et al. “Training Verifiers to Solve Math Word Problems”. In: *CoRR* abs/2110.14168 (2021). arXiv: `2110.14168 [cs.LG]`.

[Dev+19] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: `10.18653/v1/N19-1423`.

[DM46] W. J. Dixon and A. M. Mood. “The Statistical Sign Test”. In: *Journal of the American Statistical Association* 41.236 (1946), pp. 557–566. DOI: `10.1080/01621459.1946.10501898`.

[Don+25] Hongyuan Dong et al. *Scalable Vision Language Model Training via High Quality Data Curation*. 2025. arXiv: `2501.05952 [cs.CV]`.

[DZ23] Tim Dettmers and Luke Zettlemoyer. *The case for 4-bit precision: k-bit Inference Scaling Laws*. 2023. arXiv: `2212.09720 [cs.LG]`.

[Fri37] Milton Friedman. “The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance”. In: *Journal of the American Statistical Association* 32.200 (1937), pp. 675–701. DOI: `10.1080/01621459.1937.10503522`.

[Fu+22] Zihao Fu et al. *On the Effectiveness of Parameter-Efficient Fine-Tuning*. 2022. arXiv: `2211.15583 [cs.CL]`.

[Gao+24] Yunfan Gao et al. *Retrieval-Augmented Generation for Large Language Models: A Survey*. 2024. arXiv: `2312.10997 [cs.CL]`.

[Ghe24] A. Gheorghiu. *Building Data-Driven Applications with LlamaIndex: A practical guide to retrieval-augmented generation (RAG) to enhance LLM applications*. Packt Publishing, 2024. ISBN: 9781805124405. URL: `https://books.google.de/books?id=jn8CEQAAQBAJ`.

[Gur97] Kevin Gurney. *An Introduction to Neural Networks*. USA: Taylor & Francis, Inc., 1997. ISBN: 1857286731.

[HA21] Mika Hämäläinen and Khalid Alnajjar. *Human Evaluation of Creative NLG Systems: An Interdisciplinary Survey on Recent Papers*. 2021. arXiv: 2108.00308 [cs.CL].

[Hen+20] Dan Hendrycks et al. "Measuring Massive Multitask Language Understanding". In: *CoRR* abs/2009.03300 (2020). arXiv: 2009.03300.

[Hol79] Sture Holm. "A Simple Sequentially Rejective Multiple Test Procedure". In: *Scandinavian Journal of Statistics* 6.2 (1979), pp. 65–70. ISSN: 03036898, 14679469. URL: http://www.jstor.org/stable/4615733 (visited on 09/13/2025).

[Hou+19] Neil Houlsby et al. *Parameter-Efficient Transfer Learning for NLP*. 2019. arXiv: 1902.00751 [cs.LG].

[How+20] David M. Howcroft et al. "Twenty Years of Confusion in Human Evaluation: NLG Needs Evaluation Sheets and Standardised Definitions". English. In: *Proceedings of the 13th International Conference on Natural Language Generation*. Ed. by Brian Davis et al. Association for Computational Linguistics, Dec. 2020, pp. 169–182.

[Hu+21] Edward J. Hu et al. "LoRA: Low-Rank Adaptation of Large Language Models". In: *CoRR* abs/2106.09685 (2021).

[Hua+25] Lei Huang et al. "A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions". In: *ACM Trans. Inf. Syst.* 43.2 (Jan. 2025). DOI: 10.1145/3703155.

[IL24] J In and DK Lee. "Alternatives to the P value: connotations of significance". In: *Korean Journal of Anesthesiology* 77.3 (June 2024), pp. 316–325. DOI: 10.4097/kja.23630.

[Ina24] Sai Mounika Inavolu. *Exploring AI-Driven Customer Service: Evolution, Architectures, Opportunities, Challenges and Future Directions*. June 2024. DOI: 10.13140/RG.2.2.19937.31841.

[Jai22] Shashank Mohan Jain. *Introduction to Transformers for NLP. With the Hugging Face Library and Models to Solve Problems*. 1st ed. Apress Berkeley, CA, 2022, p. 165. ISBN: 978-1-4842-8843-6. DOI: 10.1007/978-1-4842-8844-3.

[Jin+24] Renren Jin et al. *A Comprehensive Evaluation of Quantization Strategies for Large Language Models*. 2024. arXiv: 2402.16775 [cs.CL].

[Jul+21] Krishna Juluru et al. "Bag-of-words technique in natural language processing: a primer for radiologists". In: *RadioGraphics* 41 (2021), pp. 1420–1426.

[Kap+20] Jared Kaplan et al. *Scaling Laws for Neural Language Models*. 2020. arXiv: 2001.08361 [cs.LG].

[Kar+20] Vladimir Karpukhin et al. “Dense Passage Retrieval for Open-Domain Question Answering”. In: *CoRR* abs/2004.04906 (2020). arXiv: 2004.04906 [cs.CL].

[Kem+16] Aniruddha Kembhavi et al. *A Diagram Is Worth A Dozen Images*. 2016. arXiv: 1603.07396 [cs.CV].

[Ker14] Dave S. Kerby. “The Simple Difference Formula: An Approach to Teaching Nonparametric Correlation1”. In: *Comprehensive Psychology* 3 (2014), 11.IT.3.1. DOI: 10.2466/11.IT.3.1. URL: https://journals.sagepub.com/doi/abs/10.2466/11.IT.3.1.

[Kri19] Klaus Krippendorff. *Content Analysis: An Introduction to Its Methodology*. Fourth Edition. Thousand Oaks, California: SAGE Publications, Inc., 2019. DOI: 10.4135/9781071878781. URL: https://methods.sagepub.com/book/mono/content-analysis-4e/toc.

[LB22] Weizhe Lin and Bill Byrne. “Retrieval Augmented Visual Question Answering with Outside Knowledge”. In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Ed. by Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 11238–11254. DOI: 10.18653/v1/2022.emnlp-main.772. URL: https://aclanthology.org/2022.emnlp-main.772/.

[Li+24] Bo Li et al. *LLaVA-OneVision: Easy Visual Task Transfer*. 2024. arXiv: 2408.03326 [cs.CV].

[Li+25] Dawei Li et al. *From Generation to Judgment: Opportunities and Challenges of LLM-as-a-judge*. 2025. arXiv: 2411.16594 [cs.AI].

[Liu+23] Nelson F. Liu et al. *Lost in the Middle: How Language Models Use Long Contexts*. 2023. arXiv: 2307.03172 [cs.CL].

[Liu+24a] Yuan Liu et al. *MMBench: Is Your Multi-modal Model an All-around Player?* 2024. arXiv: 2307.06281 [cs.CV].

[Liu+24b] Yuliang Liu et al. “OCRBench: on the hidden mystery of OCR in large multimodal models”. In: *Science China Information Sciences* 67.12 (Dec. 2024). ISSN: 1869-1919. DOI: 10.1007/s11432-024-4235-6.

[Lu+24] Shiyin Lu et al. *Ovis: Structural Embedding Alignment for Multimodal Large Language Model*. 2024. arXiv: 2405.20797 [cs.CV].

[LZ11] Yuanhua Lv and ChengXiang Zhai. “When documents are very long, BM25 fails!” In: *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’11. Beijing, China: Association for Computing Machinery, 2011, pp. 1103–1104. ISBN: 9781450307574. DOI: 10.1145/2009916.2010070.

[MBM24] Giacomo Marzi, Marco Balzano, and Davide Marchiori. "K-Alpha Calculator–Krippendorff's Alpha Calculator: A user-friendly tool for computing Krippendorff's Alpha inter-rater reliability coefficient". In: *MethodsX* 12 (2024), p. 102545. ISSN: 2215-0161. DOI: 10.1016/j.mex.2023.102545.

[Mic+25] Microsoft et al. *Phi-4-Mini Technical Report: Compact yet Powerful Multimodal Language Models via Mixture-of-LoRAs*. 2025. arXiv: 2503.01743 [cs.CL].

[Mik+13] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL].

[Min+25] Shervin Minaee et al. *Large Language Models: A Survey*. 2025. arXiv: 2402.06196 [cs.CL].

[MKJ21] Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. *DocVQA: A Dataset for VQA on Document Images*. 2021. arXiv: 2007.00398 [cs.CV].

[MM24] Priyanka Mandikal and Raymond Mooney. *Sparse Meets Dense: A Hybrid Approach to Enhance Scientific Document Retrieval*. 2024. arXiv: 2401.04055 [cs.IR].

[MP43] Warren S. McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The Bulletin of Mathematical Biophysics* 5.4 (1943), pp. 115–133. DOI: 10.1007/BF02478259.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[Nav+24] Humza Naveed et al. *A Comprehensive Overview of Large Language Models*. 2024. arXiv: 2307.06435 [cs.CL].

[NRK21] Thao Nguyen, Maithra Raghu, and Simon Kornblith. *Do Wide and Deep Networks Learn the Same Things? Uncovering How Neural Network Representations Vary with Width and Depth*. 2021. arXiv: 2010.15327 [cs.LG].

[QL16] David Quarfoot and Richard A. Levine. "How Robust Are Multirater Interrater Reliability Indices to Changes in Frequency Distribution?" In: *The American Statistician* 70.4 (2016), pp. 373–384. DOI: 10.1080/00031305.2016.1141708.

[Rad+18] Alec Radford et al. *Improving Language Understanding by Generative Pre-Training*. https://www.mikecaptain.com/resources/pdf/GPT-1.pdf. 2018.

[Ram99] Santiago Ramón y Cajal. *Texture of the Nervous System of Man and the Vertebrates. Volume I*. First English translation from Spanish, originally published 1899-1904. Springer Vienna, 1999. ISBN: 978-3-211-83057-4. DOI: 10.1007/978-3-7091-6435-8.

[Ras25] Sebastian Raschka. *Build a Large Language Model (from Scratch)*. 1st ed. From Scratch Series. New York: Manning Publications Co. LLC, 2025. ISBN: 9781638355731.

[RL24] Monica Riedler and Stefan Langer. *Beyond Text: Optimizing RAG with Multimodal Inputs for Industrial Applications*. Oct. 2024. DOI: 10.48550/arXiv.2410.21943.

[Rus+22] Sami Rusthollkarhu et al. "Managing B2B customer journeys in digital era: Four management activities with artificial intelligence-empowered tools". In: *Industrial Marketing Management* 104 (2022), pp. 241–257. ISSN: 0019-8501. DOI: 10.1016/j.indmarman.2022.04.014.

[RWW24] Jie Ruan, Wenqing Wang, and Xiaojun Wan. *Defining and Detecting Vulnerability in Human Evaluation Guidelines: A Preliminary Study Towards Reliable NLG Evaluation*. 2024. arXiv: 2406.07935 [cs.CL].

[RZ09] Stephen Robertson and Hugo Zaragoza. "The Probabilistic Relevance Framework: BM25 and Beyond". In: *Foundations and Trends® in Information Retrieval* 3.4 (2009), pp. 333–389. ISSN: 1554-0669. DOI: 10.1561/1500000019.

[Sha+24] Sanat Sharma et al. *Retrieval Augmented Generation for Domain-specific Question Answering*. 2024. arXiv: 2404.14760 [cs.CL].

[Sha+25] Minal Shah et al. *Introducing the Z-Score Normalization Technique for Hybrid Search*. Blog post. 2025. URL: https://opensearch.org/blog/introducing-the-z-score-normalization-technique-for-hybrid-search/ (visited on 07/29/2025).

[Shi+24] Yunxiao Shi et al. *Enhancing Retrieval and Managing Retrieval: A Four-Module Synergy for Improved Quality and Efficiency in RAG Systems*. 2024. arXiv: 2407.10670 [cs.CL].

[Shi+25] Divy Shikha et al. "QuickAid: A Hybrid RAG and LLM Framework for Improving Chatbot Accuracy and Relevance in First-Aid Guidance". In: *2025 International Conference on Computing and Communication Technologies (ICCCT)*. 2025, pp. 1–5. DOI: 10.1109/ICCCT63501.2025.11019682.

[Sie56] Sidney Siegel. *Nonparametric Statistics for the Behavioral Sciences*. New York: McGraw-Hill, 1956.

[SJA23] Jagdish N. Sheth, Varsha Jain, and Anupama Ambika. "The growing importance of customer-centric support services for improving customer experience". In: *Journal of Business Research* 164 (2023), p. 113943. ISSN: 0148-2963. DOI: 10.1016/j.jbusres.2023.113943.

[SKH24] Heydar Soudani, Evangelos Kanoulas, and Faegheh Hasibi. "Fine Tuning vs. Retrieval Augmented Generation for Less Popular Knowledge". In: *Proceedings of the 2024 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*. SIGIR-AP 2024. Tokyo, Japan: Association for Computing Machinery, 2024, pp. 12–22. ISBN: 9798400707247. DOI: 10.1145/3673791.3698415.

[SMK22] Ananya B. Sai, Akash Kumar Mohankumar, and Mitesh M. Khapra. "A Survey of Evaluation Metrics Used for NLG Systems". In: *ACM Comput. Surv.* 55.2 (Jan. 2022). ISSN: 0360-0300. DOI: 10.1145/3485766.

[SS17] Itay Safran and Ohad Shamir. "Depth-Width Tradeoffs in Approximating Natural Functions with Neural Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, Aug. 2017, pp. 2979–2987. URL: https://proceedings.mlr.press/v70/safran17a.html.

[Sve01] Elisabeth Svensson. "Guidelines to statistical evaluation of data from rating scales and questionnaires". In: *Journal of Rehabilitation Medicine* 33.1 (2001), pp. 47–48. DOI: 10.1080/165019701300006542.

[Szy+25] Annalisa Szymanski et al. "Limitations of the LLM-as-a-Judge Approach for Evaluating LLM Outputs in Expert Knowledge Tasks". In: *Proceedings of the 30th International Conference on Intelligent User Interfaces*. IUI '25. Association for Computing Machinery, 2025, pp. 952–966. DOI: 10.1145/3708359.3712091.

[Tea+25] Gemma Team et al. *Gemma 3 Technical Report*. 2025. arXiv: 2503.19786 [cs.CL].

[Thi25] Kailash Thiyagarajan. "Multimodal RAG for Enhanced Information Retrieval and Generation in Retail". In: *2025 International Conference on Visual Analytics and Data Visualization (ICVADV)*. 2025, pp. 102–106. DOI: 10.1109/ICVADV63329.2025.10961713.

[van+19] Chris van der Lee et al. "Best practices for the human evaluation of automatically generated text". In: *Proceedings of the 12th International Conference on Natural Language Generation*. Ed. by Kees van Deemter, Chenghua Lin, and Hiroya Takamura. Tokyo, Japan: Association for Computational Linguistics, Oct. 2019, pp. 355–368. DOI: 10.18653/v1/W19-8643.

[van+21] Chris van der Lee et al. "Human evaluation of automatically generated text: Current trends and best practice guidelines". In: *Computer Speech & Language* 67 (2021), p. 101151. ISSN: 0885-2308. DOI: 10.1016/j.csl.2020.101151.

[Vas+17] Ashish Vaswani et al. “Attention is all you need”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS’17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010. ISBN: 9781510860964.

[Wan+25] Yang Wang et al. “Evaluating Sparse and Dense Retrieval in Retrieval-Augmented Generation Systems: A Study”. In: *Proceedings of the 2024 10th International Conference on Communication and Information Processing*. ICCIP ’24. Association for Computing Machinery, 2025, pp. 548–554. ISBN: 9798400717444. DOI: 10.1145/3708657.3708747.

[Wil45] Frank Wilcoxon. “Individual Comparisons by Ranking Methods”. In: *Biometrics Bulletin* 1.6 (1945), pp. 80–83. ISSN: 00994987.

[WMC24] Yubo Wang, Xueguang Ma, and Wenhu Chen. “Augmenting Black-box LLMs with Medical Textbooks for Biomedical Question Answering”. In: *Findings of the Association for Computational Linguistics: EMNLP 2024*. Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 1754–1770. DOI: 10.18653/v1/2024.findings-emnlp.95.

[Wu+24] Zhiyu Wu et al. *DeepSeek-VL2: Mixture-of-Experts Vision-Language Models for Advanced Multimodal Understanding*. 2024. arXiv: 2412.10302 [cs.CV].

[Xio+20] Ruibin Xiong et al. “On layer normalization in the transformer architecture”. In: *Proceedings of the 37th International Conference on Machine Learning*. ICML’20. JMLR.org, 2020.

[Xio+25] Peixi Xiong et al. “Context Is All You Need: Efficient Retrieval Augmented Generation for Domain Specific AI”. In: *First Workshop on Scalable Optimization for Efficient and Adaptive Foundation Models*. 2025. URL: https://openreview.net/forum?id=6bKHoUQWlo.

[Yag23] Kevin G. Yager. “Domain-specific chatbots for science using embeddings”. In: *Digital Discovery* 2 (6 2023), pp. 1850–1861. DOI: 10.1039/D3DD00112A.

[Yue+24] Xiang Yue et al. *MMMU: A Massive Multi-discipline Multimodal Understanding and Reasoning Benchmark for Expert AGI*. 2024. arXiv: 2311.16502 [cs.CL].

[Zha+18] Xinshu Zhao et al. “We Agreed to Measure Agreement-Redefining Reliability De-justifies Krippendorff’s Alpha.” In: *China media research* 14.2 (2018).

[Zha+23] Ruochen Zhao et al. *Retrieving Multimodal Information for Augmented Generation: A Survey*. 2023. arXiv: 2303.10868 [cs.CL].

[Zhu+25] Jinguo Zhu et al. *InternVL3: Exploring Advanced Training and Test-Time Recipes for Open-Source Multimodal Models*. 2025. arXiv: `2504.10479 [cs.CV]`.

A. Appendix

A.1. Encoder comparison

To choose an encoder for dense retrieval, potential candidates were compared (see Table table:encoderComparison).

Model and **Source**	**Context Window**	**Unified Multimodal Output?**
CLIP https://huggingface.co/laion/CLIP-ViT-B-32-laion2B-s34B-b79K	77	no
jina-embeddings-v4 https://huggingface.co/jinaai/jina-embeddings-v4	32768	no
BGE-visualized https://huggingface.co/BAAI/bge-visualized	77	yes
ops-mm-embedding https://huggingface.co/OpenSearch-AI/Ops-MM-embedding-v1-2B	32768	yes
nomic-embed-multimodal-3b https://huggingface.co/nomic-ai/nomic-embed-multimodal-3b	32768	yes
llama-nemoretriever-colembed-3b-v1 https://huggingface.co/nvidia/llama-nemoretriever-colembed-3b-v1	131072	no
gme-Qwen2-VL-7B-Instruct https://huggingface.co/Alibaba-NLP/gme-Qwen2-VL-7B-Instruct	32768	yes

Table 9: Feature comparison of multimodal encoders

A.2. Selection of Generator Models

The comprehensive selection of the generator models for this study is described in full detail in this section.

A computer with 96GB of RAM, 2TB SSD and an MSI GeForce RTX 4090 with 24GB VRAM is available for this study.

Model	Language Model	Quantisations	Image Encoder	Publication Date
SAIL-VL-1d6-8B [Don+25]	Qwen-2.5-7B	BF16	AimV2-Huge	04.2024
Ovis2-16B [Lu+24]	Qwen2.5-14B-Instruct	BF16/ Int4	AimV2-Huge-patch14-448	01.2025
Ovis2-8B [Lu+24]	Qwen2.5-7B-Instruct	BF16/ Int4	AimV2-Huge-patch14-448	01.2025
InternVL3-9B [Zhu+25]	internlm3-8b-instruct	BF16/ AWQ(4bit)	InternViT-300M-448px-V2_5	04.2025
InternVL3-8B [Zhu+25]	Qwen2.5-7B	BF16/ AWQ(4bit)	InternViT-300M-448px-V2_5	04.2025
InternVL3-2B [Zhu+25]	Qwen2.5-1.5B	BF16/ AWQ(4bit)	InternViT-300M-448px-V2_5	04.2025
Qwen2.5-VL-7B [Bai+25]	Qwen2.5-7B	BF16/ FP8/ AWQ(4bit)	QwenViT	08.2024
Phi-4-multimodal [Mic+25]	Phi-4-Mini	BF16/ 4bit	SigLIP-400M	12.2024
Llava-OneVision-7B [Li+24]	Qwen2-7B	BF16	SigLIP-400M	08.2024
DeepSeek-VL2-Small [Wu+24]	DeepSeekMoE-16B	BF16	SigLIP-400M	12.2024
DeepSeek-VL2-Tiny [Wu+24]	DeepSeekMoE-3B	BF16	SigLIP-400M	12.2024
Gemma3-27B [Tea+25]	Gemma3-27B	BF16/ Int4 / 1bit / 2bit / 3bit	SigLIP-400M	03.2025
Gemma3-12B [Tea+25]	Gemma3-12B	BF16/ Int4	SigLIP-400M	03.2025
Gemma3-4B [Tea+25]	Gemma3-4B	BF16/ Int4	SigLIP-400M	03.2025

Table 10: Feature comparison of selected VLMs

The limiting factor of the hardware is the 24GB VRAM. This is closely related to the number of parameters and data type or precision of the weights of an LLM, which can be held in memory at the same time and therefore processed quickly. This raises the question whether a model with a large number of parameters and low precision per parameter or, conversely, a model with a smaller number of parameters and higher precision per parameter is preferable. In studies, Jin et al. as well as Dettmers and Zettlemoyer come to the conclusion that low precision and a higher number of parameters result in better performance as long as the precision is at least 4 bits [Jin+24] [DZ23].

In this thesis, models capable of both processing screenshots and other images as well as generating text are required. This further restricts the selection to VLMs. Table 10 compares features of selected VLMs that could potentially be considered for this study.

Model	**DocVQA val/test**	**MMMU**	**AI2D**	**MMBench 1.1**	**OCR**	**Total**
SAIL-VL-1d6-8B [Don+25]	94.9/-**	53.1	-	84	910	73.0
Ovis2-16B [Lu+24]	94.2*/-**	60.7	86.3	85.6	879	76.45
Ovis2-8B [Lu+24]	94.2/-**	57.4	86.6	83.6	891	76.45
InternVL3-9B [Zhu+25]	-/93.6	57.7	84.6	81.7	877	75.65
InternVL3-8B [Zhu+25]	-/92.7	62.7	85.2	81.7	880	77.7
InternVL3-2B [Zhu+25]	-/88.3	48.6	78.7	78.6	835	68.45
Qwen2.5-VL-7B [Bai+25]	-/95.7	58.6	-	82.6	864	77.15
Phi-4-multimodal [Mic+25]	-/93.2	55.1	82.3	77.2	844	74.15
Llava-OneVision-7B [Li+24]	90.2/87.5	48.8	81.4	80.9	622	68.15
DeepSeek-VL2-Small [Wu+24]	-/92.3	48.0	80.0	79.9	834	70.15
DeepSeek-VL2-Tiny [Wu+24]	-/88.9	40.7	71.6	70.9	809	64.8
Gemma3-27B [Tea+25]	-/86.6	64.9	84.5	79.6	753	75.75
Gemma3-12B [Tea+25]	-/87.1	59.6	75.2	74.9	702	73.35
Gemma3-4B [Tea+25]	-/75.8	48.8	74.8	66.1	660	62.3

Table 11: VLM benchmark performance comparison. - = unavailable; * = value taken from Ovis2-8B; ** = estimated as (DocVQA_val - 2.0) due to unavailable test scores.

In order to pre-select three VLMs, the performance is assessed using various benchmarks. The decision about which skills and abilities are most relevant to the use case is made based on the following analysis.

The user manuals of the knowledge base and the customer queries contain text and illustrations. Some of these illustrations are diagrams, but a much larger proportion are screenshots of the software's user interface, which is mostly table-based.

The use case of this thesis is also characterized by the fact that the domain-specific knowledge is supplemented by a RAG approach.

Therefore, the most relevant skills for answering complex customer queries are (i) logical reasoning, (ii) interpretation of tables and (iii) text recognition in high-quality images of structured documents. A high level of general knowledge and text recognition in other scenarios are less relevant. As a result, MMMU is identified for testing (i) and DocVQA for (ii) and (iii) as the most relevant benchmarks. The mean value is calculated from both scores as the final total score. Table 11 shows the performance of a selection of VLMs on different benchmarks as well as the total score.

Models with number of parameters greater 10B are not considered further (i.e. Ovis2-16B, DeepSeek-VL2-Small, Gemma3-27B, and Gemma3-12B), as they would only be able to run efficiently on the available hardware with additional measures. In strict order of descending total score, InternVL3-8B, Qwen2.5-VL-7B, Ovis2-8B, and SAIL-VL-1d6-8B, which are all based on the Qwen2.5-7B language model, would have to be selected. In order to cover more technical variation in the study, only the best-rated of these three – InternVL3-8B – is included in the preselection. In addition Phi-4-multimodal is choosen as the VLM with the next highest score and different language model. The preselection is complemented by Gemma3-4B, even though Llava-OneVision-7B and DeepSeek-VL2-Tiny score slightly better. This is due to the availability of multiple quantized variants of it's larger sister models – especially the 27B variant. With proper quantization, the 27B model has roughly the same footprint compared to the non-quantized 4B model, which makes it a promising addition to the examined trio.

A.3. Prompt Template

To specify the task for the generator models, the following prompt template was used[21]:

Role
You are an expert assistant for software development with [product name].
Task
Your task is to answer the user question. From a retrieving process, you got additional information which might help you.
ENVIRONMENT
[product name] is a software product, which is used in the context of developing software for controllers of mobile machines. The user is working with it and has questions regarding it. [product name] consists of:

- [Tool A]: This is a graphical user interface (GUI) where the system integrator defines all aspects of the machine's application. This includes managing requirements, configuring ECUs and I/O pins, defining CAN communication,

[21]Product names are anonymized and replaced by [. . .].

setting up error conditions, and managing databases and constants. The [Tool A] generates a software frame.

- [Core]: This is the essential middleware that acts as an abstraction layer between the application code and the ECU's hardware-specific Board Support Package (BSP). It provides fundamental services such as parameter handling, communication protocols (like CAN), error management, and a library of pre-tested functional blocks for I/O operations.
- [Tool B]: The [Tool B] is used for all maintenance tasks performed directly on the machine. It allows service technicians to read and write database values, view detailed information on errors, and perform software updates on the ECU.
- [Tool C]: The [Tool C] is a comprehensive tool for testing the application. It supports system tests on a virtual test bench for PC-based simulations as well as on a physical remote test bench (RTB). It also includes a Python-based scripting environment for creating automated module, integration, and application tests.
- Blocks: Predefined, tested software components designed to perform specific tasks, which helps to reduce the complexity and coding effort for an application. They are a core part of the [product name] toolboxes. These blocks serve as ready-to-use drivers and functions that can be connected to the ECU's input/output pins or used for internal signal processing

The [Core] is running on ECUs of the brand [brand name] or short [brand abbreviation]. The different types of ECUs are numbered, e.g. [Product D]. The BSP of [brand abbreviation] is deprecated when using [product name], because the [Core] adds several safety features and serves as a middleware between the BSP and the application software.

The workflow with [product name] usually works like this:

1. Define database/pins /blocks/errors/CAN-messages etc. in [Tool A]
2. Use the [Core]-functions and block-APIs and add your usecase-specific application logic.
3. When running your ECU, use [Tool B] to read errors/check pins/set and read database variables etc.
4. Write tests in [Tool C] to verify your code and the overall safety.

Process

1. Understand the environment you are working in. The user asking the question is an engineer working with the [product name] products.

2. Analyze the users question. Figure out what his goal and situation is.
3. Analyze the retrieved additional information and images.
4. Chain-of-Thought Reasoning: Does the retrieved data help solving this problem. If not, explain briefly why you can't answer the question, for example: "Sorry, I can't answer this question, because the retrieved information doesn't cover your topic. Please rephrase the question." If the retrieved data is helpful, formulate an answer to the user question. Your answer should be clear, concise and actionable. Avoid conversational filler.

Example
User Question: How can I define a CAN message in [product name] and send it cyclically?
Retrieved Additional Information: To add a CAN message navigate to ...
Images: [Image showing a GUI of [Tool A] with a button "add message".] [Image showing a GUI of [Tool A] with CAN message definitions like ID, cycle time, data signals, ...]
Output: To add a CAN message, open [Tool A] and navigate to "CAN messages". Click the button "add message", a window will open. Insert CAN ID and other information. Choose a buffer and set the message type to "TXC" for cyclic sending.
Your Turn
User Question:

A.4. Stemming Test

To verify that the stemming process does not negatively impact clarity and uniqueness, a test with selected domain-specific terms and abbreviations was performed. The following console printout shows that the cleaned version preserves all information which makes the words distinct. Although additional whitespace was inserted between lines of code during the cleaning process, this should not effect the retrieval process negatively.

```
Original:
u16Cnt u16TimeoutSignal DM_AppIsBusy PWM high-side
ERR_CAT0_NOT adc eCoreAdcGetPinFreshFb
OutPwm gCsr_tFCCU_Conf1_Mux0.u8Signal
eBloCSndNow(&gCSnd_tFCCU_Conf);
J1939 DM1
Proportional Output Block
//! @param[in]  ptSigAdr        - [stu] LUT signal address


Cleaned:
u16cnt u16timeoutsign dm_appisbusi pwm high-sid
```

```
err_cat0_not adc ecoreadcgetpinfreshfb
outpwm gcsr_tfccu_conf1_mux0.u8sign
eblocsndnow ( & gcsnd_tfccu_conf ) ;
j1939 dm1
proport output block
// ! @ param [ in ] ptsigadr - [ stu ] lut signal address
```

A.5. Mapping of Request IDs

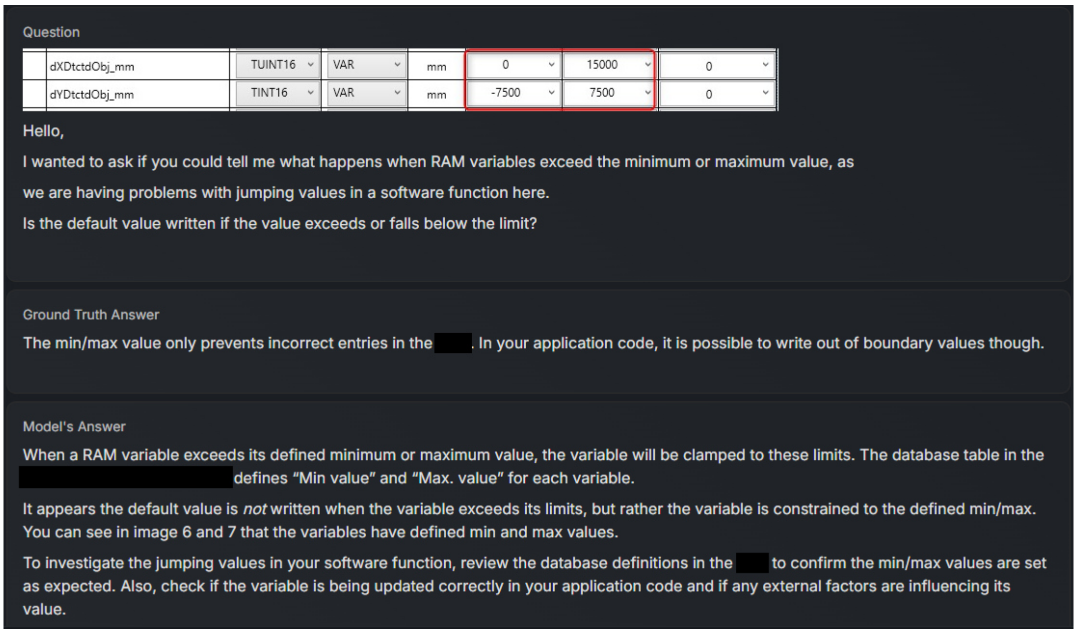
To improve readability, the requests referred to in the discussion (Section 7) have been assigned a new, consecutive numbering system. For full transparency and data traceability, the mapping to the original IDs is provided in Table 12.

Identifier Used in the Discussion	Original Request ID
1	51274
2	50303
3	110362
4	110130
5	98124
6	47777
7	101790
8	68612
9	44855
10	82653
11	95692

Table 12: Mapping of new request identifiers to original IDs

A.6. Argilla

For the annotation process, Argilla was used. Figure 17 shows the page structure with easy comparison between ground truth and model answer. Annotators could access the definitions of each ordinal value conveniently as shown in Figure 18.


Question

	dXDtctdObj_mm	TUINT16	VAR	mm	0	15000	0
	dYDtctdObj_mm	TINT16	VAR	mm	-7500	7500	0

Hello,

I wanted to ask if you could tell me what happens when RAM variables exceed the minimum or maximum value, as

we are having problems with jumping values in a software function here.

Is the default value written if the value exceeds or falls below the limit?

Ground Truth Answer

The min/max value only prevents incorrect entries in the [redacted]. In your application code, it is possible to write out of boundary values though.

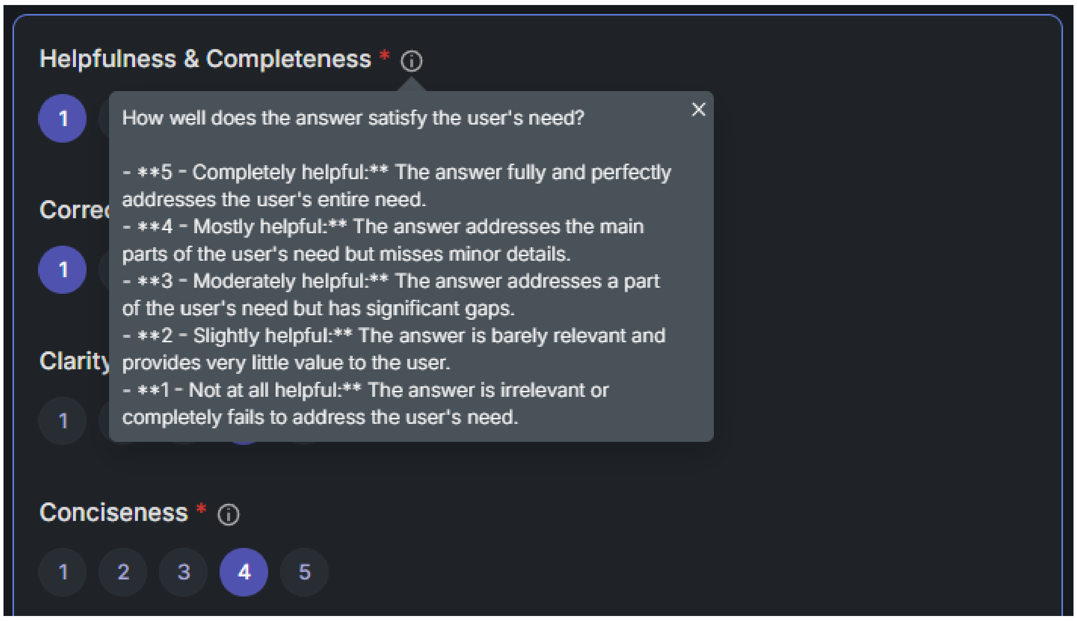
Model's Answer

When a RAM variable exceeds its defined minimum or maximum value, the variable will be clamped to these limits. The database table in the [redacted] defines "Min value" and "Max. value" for each variable.

It appears the default value is *not* written when the variable exceeds its limits, but rather the variable is constrained to the defined min/max. You can see in image 6 and 7 that the variables have defined min and max values.

To investigate the jumping values in your software function, review the database definitions in the [redacted] to confirm the min/max values are set as expected. Also, check if the variable is being updated correctly in your application code and if any external factors are influencing its value.

Figure 17: Argilla GUI showing Question, Ground Truth and Model's Answer. Names of tools are anonymized.


Helpfulness & Completeness *

How well does the answer satisfy the user's need?

- **5 - Completely helpful:** The answer fully and perfectly addresses the user's entire need.
- **4 - Mostly helpful:** The answer addresses the main parts of the user's need but misses minor details.
- **3 - Moderately helpful:** The answer addresses a part of the user's need but has significant gaps.
- **2 - Slightly helpful:** The answer is barely relevant and provides very little value to the user.
- **1 - Not at all helpful:** The answer is irrelevant or completely fails to address the user's need.

Corre

Clarity

Conciseness *

1 2 3 [✓ 4] 5

Figure 18: Argilla GUI showing annotation categories with additional explanation

A.7. Qualitative Retrieval Analysis

A qualitative analysis of samples of helpfulness scores 1 revealed the failure of retrieval as the dominant cause of failure. Table 13 shows (i) that the retrievers failed to detect the similarity between query and relevant chunk or (ii) that even a human expert was not able to find the relevant information in the database.

ID	Keywords of Ground Truth Answer	Relevant Chunk
49052	InFreq block; output structure	ItfBloInFreq_chunk_1: "[. . .] TUint16 u16Cnt; //!< [cnt] Current count value [. . .] TInFreqOut [. . .]"
50041	InFreq block; timeout for getting signal	ItfBloInFreq_chunk_0: "[. . .] TUint16 u16TimeoutSignal; //!< [ms] - Timeout of input signal [. . .]"
86834	BloErr, vBloErrSetInpBit(), Application Error List, Error Detection Method	C_UserManualDoc6_chunk_365: Beispielanwendung, "[. . .] vBloErrSetInpBit(&gErr_tApp_ERR_01, DM_AppIsBusy, FALSE) [. . .]"
89429	Dither Frequency	-
96243	BloOutPwm	ItfBloOutPwm_chunk_0: "[. . .] //!The Block is designed to control a high side pulse-width modulation (PWM) output. //!The Block gets desired duty cycle from input and sets it to the output pin. [. . .]"
97342	BloDig, High-side, Low-side	ItfBloDig_chunk_0: "[. . .] state of high-side [. . .]"
98054	ERR_CAT0_NOT, Error Severity	ALL-UsrMan_[ToolA]_2_12_91_0278_chunk_252: "[. . .] Set the property Error Severity to a different value than ERR_CAT0_NOT [. . .]"
102816	Can Receive Block; output structure	-
113563	eCoreAdcGetPinFreshFb()	ItfCorePin_chunk_2: "[. . .] Request of a adc value e.g. ADC_DOU_C [. . .] EPinStatus eCoreAdcGetPinFreshFb [. . .]"

Table 13: User requests where all retrieval strategies failed to find the relevant chunk, resulting in the lowest helpfulness score of 1. Manual verification confirmed the chunk's existence within the database by searching based on keywords derived from the ground-truth answer.