Universiteit Leiden
The Netherlands

Bachelor Computer Science & Economics

Enhancing RAG-Based Question Answering: A Case Study on Chunking, Prompt Engineering, and Automated Evaluation at Leadinfo

Reinier Matthias Alexander Bos

Supervisors:

Dr. Peter van der Putten & Dr. Joost Broekens

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)

www.liacs.leidenuniv.nl 01/07/2025

Abstract

Large language models (LLMs) often hallucinate and lack company-specific knowledge, which limits their value in business settings. Retrieval-augmented generation (RAG) can be a solution by combining large language models with document retrieval to answer domain-specific questions. RAG's effectiveness depends on design choices like document chunking and prompt formulation. This study compares two chunking methods—fixed-size and semantic—and examines three prompt engineering strategies—keyword expansion, query reformulation, and chain-of-thought prompting—using a RAG chatbot developed for Leadinfo. It also compares human and LLM evaluations. The performance is assessed by both humans and an LLM focusing on faithfulness, answer relevance, and context relevance. The results show that fixed-size chunking performs as well as semantic chunking across all metrics. However, prompt-engineering strategies significantly lowered evaluation scores by $\sim$18% on average. Agreement between human and LLM raters was only slight, and the LLM scored answers $\sim$0.8 points higher on a 5-point scale. They do agree substantially on multiple-choice questions. These results suggest that simple chunking combined with minimal prompting produces the best outcomes. For evaluation, a hybrid approach provides the best balance of cost and reliability. Humans assess open questions and generate a ground-truth set that the LLM can use for large-scale automated evaluation. Future research should improve hybrid workflows to close the gap between automated and human judgment in real-world, complex applications.

Contents

1 Introduction

Large language models (LLMs) such as GPT-4 have become increasingly popular for knowledge work [AAA+23]. However, organizations still struggle to obtain reliable, domain-specific answers. That is because proprietary information is missing from public training data, and the models' knowledge freezes at the moment of training, making it quickly outdated [CHGD24]. These limitations restrict the usefulness of standard LLM solutions in commercial environments where accuracy and confidentiality matter.

Retrieval-Augmented Generation (RAG) offers a solution by letting an LLM ground its answer in documents retrieved from an internal knowledge base. While promising, the effectiveness of a RAG system depends on many design choices. In particular, how documents are split into chunks and how prompts are engineered may strongly influence both retrieval accuracy and answer quality, yet their combined impact remains underexplored. This research will look into these design choices.

Using RAG does not guarantee good results. Therefore, it is important to evaluate the quality of a RAG system. As RAG systems give open responses, which can be formulated in many ways, it is hard to evaluate automatically, but human evaluation is expensive and time-consuming. There is a growing demand for RAG applications, and thus for automated evaluation. Therefore, this research will look into how human evaluation compares to automated LLM evaluation.

1.1 Research Question

As there are different ways to design and evaluate a RAG chatbot, the following research question emerges:

"How do chunking techniques and prompt engineering strategies impact RAG-based question answering, and how does LLM-based evaluation compare to human judgment in assessing RAG performance?"

1.2 Thesis Overview

The remainder of this thesis is structured as follows: Section 2 describes the relevant background information for this thesis; Section 3 discusses the approach of this research; Section 5 describes the experiments and their outcome; Section 6 evaluates the results. Section 7 concludes the thesis.

This bachelor's thesis was written as part of the Computer Science & Economics program at LIACS (Leiden Institute of Advanced Computer Science), in collaboration with Leadinfo, and under the supervision of Dr. Peter van der Putten and Dr. Joost Broekens.

2 Background

This section will cover the background literature on LLMs and RAG, particularly chunking strategies and prompt engineering techniques. Different human and automated LLM evaluation methods will also be covered.

2.1 Large Language Models and Their Limitations

Deploying Large Language Models (LLMs) in business settings remains challenging for several reasons [AAA+23]. First, because proprietary or confidential information is absent from public training corpora, LLMs cannot provide domain-specific answers. Second, their knowledge is frozen at training time, so responses can be outdated or the LLM can hallucinate [FDN+24]. Finally, worries over data privacy, model bias, and the "black-box" nature of current systems further hinder adoption in industry [CHGD24]. These limitations restrict the usefulness of off-the-shelf LLMs in commercial applications.

2.2 Retrieval-Augmented Generation

A possible solution to this problem is Retrieval-Augmented Generation (RAG). RAG is an approach that combines retrieval-based techniques with generative models to improve the quality and accuracy of responses. Figure 1 shows how a RAG chatbot works [Yin24].

1. The first step is to build this database with relevant and up-to-date information. The data can come from different sources, like internal documents, PDFs, or a website.

2. The embedding model encodes the documents into embeddings. The document embeddings are stored in a vector database. This allows for fast semantic similarity search later on.

3. The user asks a question in natural language, which is also stored using the same embedding model, so it can be compared with the stored documents.

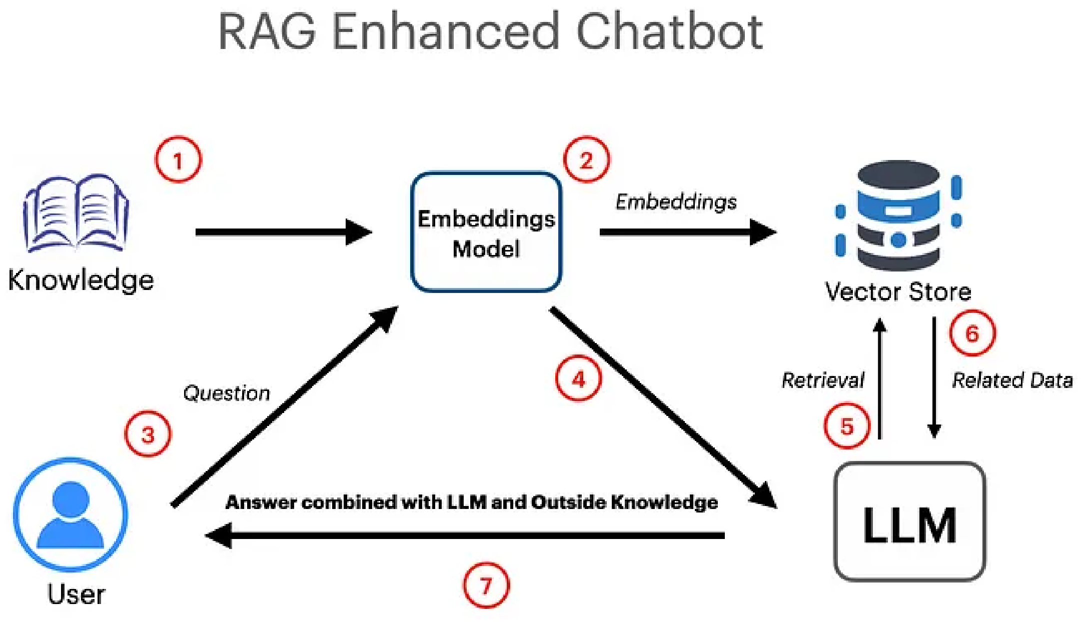


Figure 1: RAG pipeline with retrieval and generation modules.

- Optional: Apply prompt-engineering by, for example, adding key domain terms or clarifying vague wording, so the query carries enough context and is less ambiguous before it is embedded.

4. The embedded user question is used to search the vector store. The most relevant documents or chunks are retrieved based on semantic similarity.

5. The LLM combines the (engineered) user's question and the retrieved documents to generate a response.

6. The answer, based on grounded information, is sent to the user.

These steps ensure that a RAG chatbot provides the user with an answer based on relevant data. However, multiple aspects determine the quality of output of a RAG chatbot. Two of these factors are:

1. Chunking strategy: Data from a knowledge base is divided into smaller chunks, since LLMs can handle only a limited amount of context [L+24]. In addition, a larger context comes with larger financial costs. It is also not desirable to provide the user with too much information.

2. Prompt quality: The performance of an LLM depends on the quality of a prompt. Especially in a RAG chatbot, the correct documents must be retrieved [ZHS+24]. Users can provide a chatbot with a wide variety of prompts, which vary a lot in quality as well. To get more stable and improved responses, the prompt can be engineered to get better results. To make answers more consistent and relevant, the prompt can be enriched with user-specific data, such as the user's department, role, or current project, so that the RAG chatbot retrieves and ranks documents that best match that context.

2.2.1 Chunking Strategies in RAG

The chunking strategy to be used is an often overlooked part of a RAG system [ZJF+24]. Chunking plays a crucial role in RAG, as it determines what information will be added to the context of the prompt. If the data is not chunked well, it can lead to incomplete contexts or excessive irrelevant information, which impacts the performance. Additionally, a larger context comes with larger computational costs. Therefore, it is important to have a well-designed chunking strategy. Multiple chunking strategies can be applied, such as fixed-size chunking, paragraph-based chunking, semantic chunking, markdown-based chunking, and Perplexity chunking. Table 1 shows an overview of the different strategies.

Fixed-size chunking means that the data is split into chunks of a fixed size (e.g. 1000 tokens) [LPP+20]. These chunks can be disjoint or overlapping (e.g., 200 tokens). Most often, the chunks are overlapping to help preserve context and make sure an important answer is not split between two chunks. The advantage of this strategy is that it is very fast and requires very limited computation. The disadvantage is that this breaks logical units, because it can split data in the middle of a sentence or paragraph. It works well for factual data, but less for data where coherence is critical.

Paragraph-based chunking splits text on natural language barriers like the end of a sentence or paragraph [WNM+19]. This method respects logical units, which improves coherence in chunks.

Strategy	**Description**	**Advantages**	**Disadvantages**
Fixed-size chunking	Splits text into fixed-size chunks (e.g., 1000 tokens) regardless of meaning.	Fast and easy to implement. Low computational cost.	Breaks logical units. Poor coherence.
Paragraph-based chunking	Splits at natural language boundaries such as sentences or paragraphs.	Respects logical structure. Improves coherence.	Leads to imbalance in chunk sizes. Affects embedding consistency.
Markdown-based chunking	Splits content based on document structure such as headers, sections, or HTML tags.	Ideal for structured documents (FAQs, manuals). Preserves hierarchy.	Not suitable for unstructured or plain text.
Semantic chunking	Splits based on changes in meaning or topic, often using embeddings or similarity.	Keeps semantically related content together. Improves retrieval relevance.	Computationally intensive. Requires embeddings or similarity models.
Perplexity chunking	Uses language model perplexity to detect topic shifts and define chunk boundaries.	Adapts to logical/language transitions. Improves coherence.	Requires language model scoring. Slightly slower than rule-based methods.

Table 1: Comparison of chunking strategies for RAG Systems.

However, it can also lead to a loss of coherence between chunks. Another disadvantage is that it leads to an imbalance in chunk sizes. As each chunk is embedded with a vector, it will lose nuance for longer or shorter chunks, which leads to uneven retrieval quality.

Markdown-based chunking uses the document structure, like markdown headers, HTML tags or titles, to split content [GXG+23]. This works great for structured documents like FAQs or manuals. It does not work well for raw text, as there is no clear document structure.

Semantic chunking splits the text based on semantic coherence [GXG+23]. This means that a new chunk is started when the similarity with the previous text drops. Therefore, related information is kept together within each chunk. This approach improves the quality of retrieval in RAG systems, but it is more computationally intensive.

Perplexity chunking is a chunking strategy proposed by [ZJF+24]. This method addressed the limitations of traditional chunking strategies. It identifies and groups sentences with linguistic and logical connections. Perplexity chunking works by analyzing the distribution of perplexity scores across sentences. A perplexity score represents how well a language model predicts a given piece of text. The lower the perplexity, the more confident a model is in predicting the next words. If there is a significant change, the text is segmented into chunks.

2.2.2 Prompt Engineering Techniques

Prompt engineering is another part of a RAG system that can have a big impact on performance [SSS+24]. When everything part of the RAG pipeline is set up well, but the prompt is not well understood, it will not lead to good performance. A too short, vague, or under-specified prompt can lead to incomplete retrieval and incorrect or irrelevant answers. Expanding the prompt not only has benefits, but a poorly expanded prompt can unnecessarily increase the context length. Therefore, it is important to have a well-designed prompt engineering strategy. Several prompt engineering techniques can be applied, such as keyword expansion, query reformulation, few-shot expansion, and chain-of-thought prompting. Table 2 shows an overview of the different techniques.

Keyword expansion adds relevant keywords or synonyms to the original query [ZW24]. Instead of only searching based on the keywords present in the original prompt, the prompt can be enhanced to contain more synonyms or relevant keywords. The advantage of this strategy is that it increases recall during retrieval by matching more documents. The risk of this technique is that the prompt is altered too much and that it introduces irrelevant information.

Another strategy is query reformulation [SZ24]. That is, to rewrite the user query in a more detailed or unambiguous form. The method improves the specificity of the retrieval phase, ensuring better precision. However, the user's query may be misunderstood. It is especially useful for short, vague queries. It can also be expanded by injecting user context, which enriches the query by appending relevant background information before retrieval. This could be information like the department or previous queries.

Few-shot expansion involves providing the model with a few examples of similar question-answer pairs. This helps the model better understand the type of response expected. While this improves model robustness, it also consumes more prompt space, which can become an issue for longer documents.

Chain-of-Thought prompting guides the model to reason step-by-step before answering. This can improve factuality and logical consistency in complex queries. However, it can increase response latency, as the model generates longer outputs.

2.3 Evaluation of RAG Systems

RAG systems offer many possibilities for customization, as there are, for example, numerous options for chunking and prompt engineering. This leads to the question of which configuration is most effective in a specific domain. This is referred to mainly as RAG evaluation [BSB25]. The evaluation of RAG consists of systematically evaluating the performance of a RAG system. This remains a complex challenge, as traditional evaluation metrics like exact match or F1-score are not suited for open-ended or generated responses, because these answers can be partially correct, contextually dependent, or phrased differently. Therefore, different evaluation techniques have emerged, focusing on human judgment and increasingly on LLMs as evaluators [BSB25], [SFKPZ23]. A crucial aspect of the evaluation is assessing the performance of the retrieval component [BSB25]. This means determining whether the retrieved chunks are relevant to the query. RAG systems are often evaluated on the following aspects [YGZ+24]:

- **Faithfulness**: How accurate is the answer generated by the system based on the context?
- **Answer relevance**: How relevant is the answer to the given query?
- **Context relevance**: How relevant is the retrieved context to the query?

Technique	**Description**	**Advantages**	**Disadvantages**
Keyword expansion	Adds synonyms or related terms to the query.	Increases recall. Improves document matching.	Risk of introducing irrelevant terms.
Query reformulation and context injection	Rewrites the user query in a more detailed or unambiguous form, potentially enriched with user context such as department, location, or previous queries.	Improves retrieval precision. Personalizes and clarifies the query.	Possible misinterpretation of user intent. Risk of bloated or noisy queries.
Few-shot expansion	Provides few-shot examples in the prompt to guide the model.	Boosts model robustness and improves understanding of expected answers.	Increases prompt length and computational cost.
Chain-of-Thought prompting	Guides the model to reason step-by-step before answering.	Improves factuality, reasoning quality, and complex answer consistency.	Longer response times. Higher token usage.

Table 2: Comparison of prompt engineering strategies for RAG systems.

2.3.1 Types of Evaluation

Human judgment of a RAG system can offer high-quality insights [ZLX+24]. Especially in a domain-specific scenario, human insights are valuable. However, human judgment is time-consuming, expensive, and can be subject to bias. Therefore, there is a growing need for automated judgment [BSB25]. Many factors influence the outcome, making it difficult to automate. Several methods, such as RAGAS [EJAS24] and ARES [SFKPZ23], have been proposed to evaluate RAG systems. RAGAS evaluates systems on the metrics context relevance, answer faithfulness, and answer relevance, while ARES is focused on hallucination detection.

2.3.2 Reference Database

A reference database is crucial to evaluate the performance [BSB25]. These databases contain question-answer pairs, which can be used as a reference to judge the performance of a RAG pipeline. The database can be created by humans. For example, by human experts in a specific domain. An existing dataset can also be enhanced. This can be done for a database that contains questions and context, but no answers [XCJS24]. Or, a database can be enhanced to reflect specific error types to assess their robustness and reliability [PNEF24]. A dataset can also be completely created using an LLM, as the creation of a database is time-consuming. By providing an LLM with context, it can generate question-answer pairs related to that context [TY24]. To create a diverse dataset, different types of questions can be generated. Different evaluation scenarios can be used, for example, the scenarios proposed by RAGProbe [SBK+24]:

- **S1**: A question to retrieve a number for which the answer is in a single document

- **S2**: A question to retrieve the date/time for which the answer is in a single document
- **S3**: A multiple-choice question for which the answer is in a single document
- **S4**: A question combining multiple questions for which the answers are in a single document
- **S5**: A question combining multiple questions for which the answers are in a set of documents
- **S6**: A question for which the answer is not in the document corpus

Another framework that characterizes question difficulty and cognitive complexity is Bloom's taxonomy [For10]. Bloom's taxonomy ranks cognitive skills into a hierarchy. This ranges from simple recalling to more difficult skills such as analyzing, evaluating, or creating. This hierarchy can be used to create a diverse set of questions for the reference database. Bloom's taxonomy has the following levels:

- **Remember:** Recall facts and basic concepts
- **Understand:** Explain ideas or concepts
- **Apply:** Use information in new situations
- **Analyze:** Draw connections among ideas
- **Evaluate:** Justify a decision or course of action
- **Create:** Produce new or original work

2.4 Research Gap and Motivation

Retrieval-Augmented Generation (RAG) is a strong approach to enhance Large Language models and overcome their limitations, especially in a commercial setting. Multiple things affect the quality of retrieved documents and the response. Previous research has shown that chunking strategies play a crucial role in retrieval effectiveness. Fixed-size chunking, paragraph-based chunking, semantic chunking, markdown-based chunking, and perplexity-based chunking have all been explored to different extents. Additionally, prompt engineering techniques have been studied to improve how user queries interact with retrieval and generation. Different methods like keyword expansion, query reformulation, context injection, few-shot prompting, and chain-of-thought prompting have been shown to increase retrieval precision and response quality.

Prior research on the separate effects of chunking strategies and prompt engineering. There is a lack of studies that systematically examine the combined effect of chunking strategies and prompt engineering. Multiple studies evaluate the methods in academic benchmarks, but this differs as it is performed in a business setting.

The main gap is a lack of standardized and robust evaluation approaches for RAG systems, especially in applied settings. Traditional evaluation metrics, like Exact Match of F1-score, are not suitable for open-ended answers. As a result, other methods have to be used, like human judgment and the use of LLMs as evaluators. However, human judgment is time-consuming and expensive, but the quality of the LLM evaluation is not guaranteed. Therefore, research is needed to create human-like automated evaluation for RAG systems.

These gaps highlight the need for an approach that combines chunking and prompt engineering techniques and assesses the reliability and scalability of different evaluation methods of this approach. By addressing this gap, the research contributes to an understanding of the interaction effects of chunking and prompting strategies. It also offers insight into different evaluation methods of RAG systems.

3 Methodology

This chapter explains how the study was designed and conducted. It covers the data-gathering process, the design of the RAG chatbot, and the procedures used to compare human and LLM evaluations. Details of the RAG chatbot and automated evaluation implementation are discussed in Chapter 4.

3.1 Research Design

This research follows the Design Science Research methodology. A baseline RAG chatbot was developed and evaluated. The implementation and evaluation were done at Leadinfo [Lea25a]. The code of the RAG chatbot and evaluation can be found on GitHub. The chatbot focused on Leadinfo's product information and was targeted at employees of the commercial department. Different RAG pipelines were configured and tested. Retrieval and response quality were evaluated by humans and an LLM.

3.2 Dataset and Preprocessing

The RAG chatbot worked with a database that contains Leadinfo data. Data was retrieved from different sources, namely the Leadinfo Help Center [Lea25b], an internal Notion database [Not25], and the Leadinfo website [Lea25a]. The Help Center contains different articles explaining different aspects of Leadinfo. The Notion database contains the following elements:

- **Competitors**: A list of Leadinfo's competitors containing relevant information about them, and things where Leadinfo outperforms competitors.
- **Product FAQs**: A list of FAQs about Leadinfo.
- **Content center**: A list of materials for customers or partners of Leadinfo.
- **Product feedback**: A list containing product feedback on Leadinfo.

All of this data was cleaned up and stored with the containing metadata, so that sources could be provided.

3.3 RAG Chatbot

A RAG chatbot was developed to execute the experiment. Different RAG pipelines were constructed to evaluate the impact of design choices, especially regarding chunking strategies and prompt engineering techniques. The implementation details are described in Chapter 4.

3.3.1 Chunking Strategies

In Section 2.2.1, different chunking strategies have been discussed. For this research, fixed-size chunking and semantic chunking were compared. Fixed-size chunking has the advantage of being low-cost, while semantic chunking can preserve content better. The other strategies mentioned, paragraph-based chunking, markdown-based chunking, and perplexity chunking, were not selected. Paragraph-based chunking can result in highly imbalanced chunk sizes and inconsistent embeddings, which hurts retrieval. Markdown-based chunking is mainly suitable for highly structured documents; however, the documents in this research are not all structured. Finally, perplexity chunking requires extra complexity due to the need for language model scoring and tuning, which is not in the scope of this study.

3.3.2 Prompt Engineering Techniques

In Section 2.2.2, several prompt engineering techniques have been covered. In this experiment, a baseline version without prompt engineering was compared to an enhanced version that applies keyword expansion, query reformulation, and chain-of-thought prompting. These methods were selected because they impact the quality of document retrieval and the clarity of generated responses. Keyword expansion is used because it should improve recall. Query reformulation should improve precision, and context injection incorporates relevant background information, such as previous messages, to improve disambiguation. Chain-of-thought prompting is included, so that the model reasons step-by-step. This should improve the answer clarity, especially when the queries are more complex. Few-shot prompting is excluded because it can bloat the prompt with too much context.

3.3.3 RAG Pipelines Setup

To assess the impact of chunking and prompt engineering strategies, four RAG pipeline configurations are implemented:

1. **Model 1 (Baseline)**: Fixed-size chunking, no prompt engineering.
2. **Model 2 (Chunking-enhanced)**: Semantic chunking, no prompt engineering.
3. **Model 3 (Prompt-enhanced)**: Fixed-size chunking, with prompt engineering.
4. **Model 4 (Combined-enhanced)**: Semantic chunking, with prompt engineering.

3.4 RAG Evaluation

The performance of the RAG chatbot and the effectiveness of different chunking and prompt engineering strategies were evaluated on the following aspects using a Likert scale [BH17].

- **Faithfulness**: How accurate is the answer generated by the system based on the context?
- **Answer relevance**: How relevant is the answer to the given query?
- **Context relevance**: How relevant is the retrieved context to the query?

These metrics were used to compare the four proposed model versions and to draw conclusions about the impact of chunking and prompt engineering techniques.

3.4.1 MCQ Database

To obtain an objective ground truth while preserving the open-ended nature of RAG answers, a structured multiple-choice-question (MCQ) database was created. Each item consisted of a generated question, the correct option, and three false answer options. The MCQ enabled measuring how accurately an LLM judge selected the correct option, and measuring agreement between LLM and human raters.

The LLM was supplied with the entire Leadinfo database and instructed to cluster related articles so that questions could draw on information from multiple related sources. Following the scenarios defined in RAGProbe, adapted to this use case [SBK+24]. The six scenarios are combined into four by treating token type (number vs date) and question format (MCQ vs free text) as superficial. That means S1-S3 are merged, while keeping S4 for single-document synthesis, S5 for cross-document synthesis, and S6 as the unanswerable control.

Each question will also be created based on one of the first four levels of Bloom's taxonomy: remember, understand, apply, or analyze. The two higher Bloom levels were excluded because they do not work well with the multiple-choice format.

The answers were also compared with cosine similarity applied to TF-IDF vectors [WD20]. This is a quick and affordable way to compare two outputs. This fast, low-cost metric made it possible to compare human and LLM evaluation with a baseline.

3.4.2 Human Evaluation

To assess the quality of the chatbot's responses in a reliable and interpretable manner, several domain experts evaluated the chatbot. They evaluated the chatbot's answers using a Likert scale on the aspects of faithfulness, answer relevance, and context relevance. Additionally, the evaluators received a question, with a generated answer by the RAG chatbot, together with the answer options from the multiple-choice database. They chose the answer option that is most semantically similar to the generated response.

The human evaluators were instructed on the definition and guidelines for each evaluation dimension to ensure consistency. The inter-rater agreement was assessed with Cohen's kappa [VKS10]. The human evaluation provided insight into the performance of different RAG pipelines and was used to compare with the automated evaluation.

3.4.3 Automated LLM Evaluation

The evaluation was also done automatically with an LLM as a judge. The LLM was presented with the same query and generated answer as humans. However, it received the whole file as context, instead of just the chunks, as that would mitigate the impact of different chunking strategies. Using this information, it rated the answers on the same aspects as humans with a Likert scale. The LLM also evaluated the MCQ database by selecting the answer option most semantically similar to each generated response.

This automated evaluation enabled large-scale benchmarking with minimal human effort and provided a consistent metric between model versions. The judgments were compared to human judgment. This provided insight into the ability of an LLM to evaluate the performance of a RAG pipeline while also highlighting how it performed across varying question difficulties and Bloom levels.

3.5 Experiment Setup

A structured experiment was conducted at Leadinfo to evaluate the RAG pipelines and to compare human and automated assessments. Six Leadinfo employees (domain experts) first posed free-form, domain-specific questions to the RAG chatbot, yielding 64 question-answer pairs. Each answer was then rated on a Likert scale for faithfulness, answer relevance, and context relevance by the user who asked it and by an LLM. In a second phase the same raters, plus the GPT-4 judge, viewed the chatbot's answer alongside four multiple-choice options, which the chatbot itself had never seen, and selected the option whose meaning best matched the answer. Finally, as a naive baseline, the cosine similarity between the answer's embedding and each option's embedding was computed, and the top-scoring option was taken as the baseline prediction.

4 Implementation Details

To conduct the experiments, a RAG chatbot was implemented. The chatbot system was designed to retrieve relevant documents from an internal knowledge base and generate context-aware answers using a Large Language Model.

4.1 Chatbot Framework

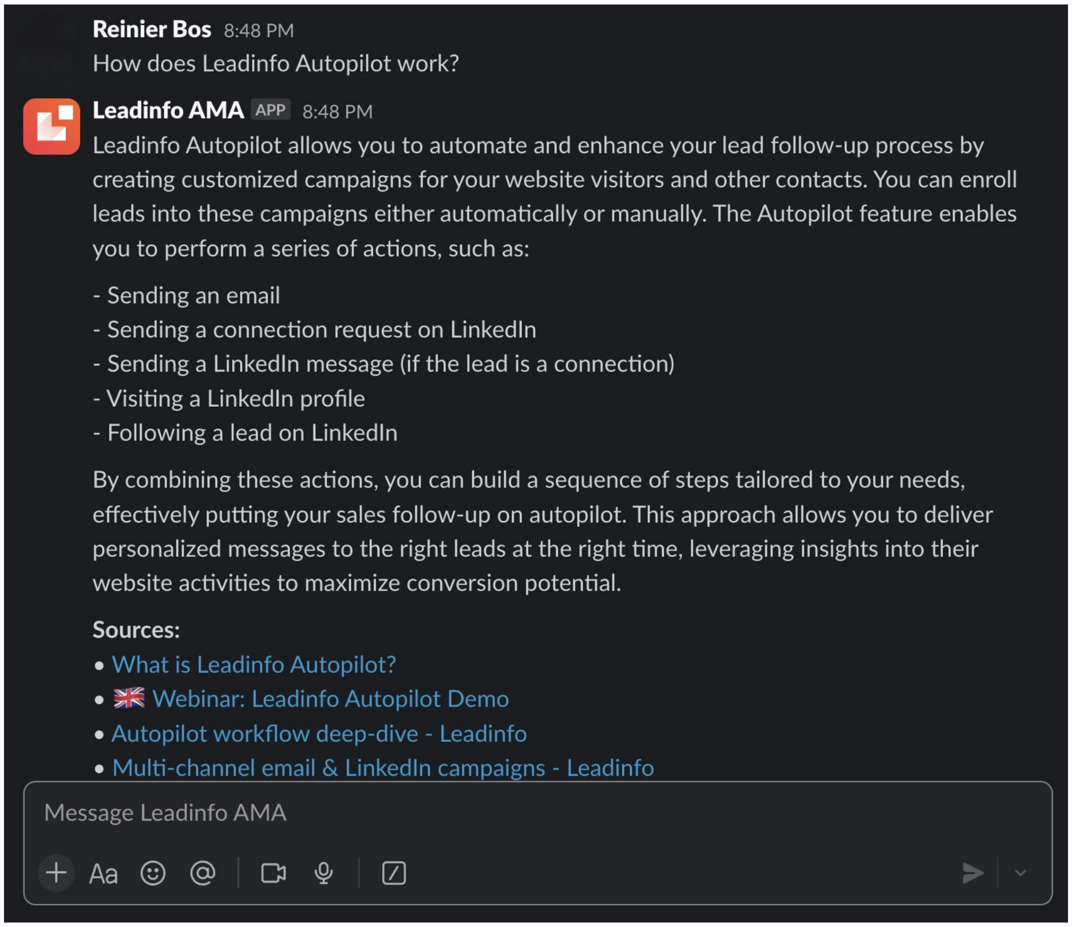
The chatbot was built using different components. The database consisted of a set of text files, retrieved from Leadinfo's website, Leadinfo's help center website, and Notion pages. The database was embedded in a vector database with FAISS, which is an open-source vector database, designed for efficient similarity search and retrieval [DGD+24]. The RAG pipelines were created with the open-source LangChain framework [TA23]. LangChain is suitable for prompt engineering. The LLM selected for answer generation is GPT-4o, which is a fast, intelligent, and flexible model, with a 128k tokens context window, which is suitable for understanding context well [Ope24].

The RAG chatbot is developed for Leadinfo. The use case for the chatbot is for employees to be able to get answers fast on different aspects of Leadinfo. That is done because Leadinfo is a fast-growing company, so there are many employees with limited experience. Having a chatbot that can provide answers to most questions reduces the load on experienced colleagues, who would normally receive the questions. For this use case, the RAG chatbot must be easily accessible for employees. Therefore, it is integrated with Slack, which is a messaging platform for teams and companies [Sla25a]. The platform is used at Leadinfo, and by integrating the RAG chatbot here, it is easily available to all colleagues. To do this, an existing framework for an AI chatbot in Slack is used [Sla25b]. This is modified to fit the needs, and the RAG pipelines are integrated into this platform. Figure 2 shows an example question asked of the RAG chatbot in Slack.

4.1.1 Chunking Implementation

For fixed-size chunking, the text was split into overlapping segments of 1000 tokens with an overlap of 200 tokens. This is shown in Figure 3.

Semantic chunking was implemented using a cosine similarity threshold between sentence embeddings generated by a SentenceTransformer model. A new chunk was started whenever the semantic similarity between sentences dropped below a predefined threshold. This is shown in Figure 4


Reinier Bos 8:48 PM
How does Leadinfo Autopilot work?

Leadinfo AMA APP 8:48 PM
Leadinfo Autopilot allows you to automate and enhance your lead follow-up process by creating customized campaigns for your website visitors and other contacts. You can enroll leads into these campaigns either automatically or manually. The Autopilot feature enables you to perform a series of actions, such as:

- Sending an email
- Sending a connection request on LinkedIn
- Sending a LinkedIn message (if the lead is a connection)
- Visiting a LinkedIn profile
- Following a lead on LinkedIn

By combining these actions, you can build a sequence of steps tailored to your needs, effectively putting your sales follow-up on autopilot. This approach allows you to deliver personalized messages to the right leads at the right time, leveraging insights into their website activities to maximize conversion potential.

Sources:
- What is Leadinfo Autopilot?
- 🇬🇧 Webinar: Leadinfo Autopilot Demo
- Autopilot workflow deep-dive - Leadinfo
- Multi-channel email & LinkedIn campaigns - Leadinfo

Figure 2: Example interaction with the RAG chatbot in Slack.

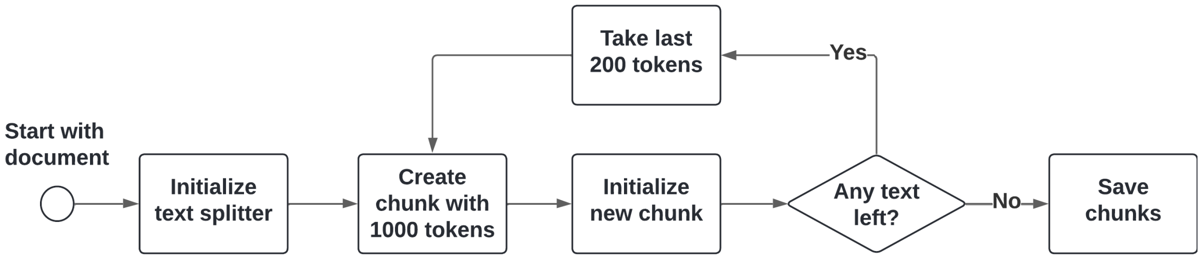


Figure 3: Flow chart of fixed-size chunking.

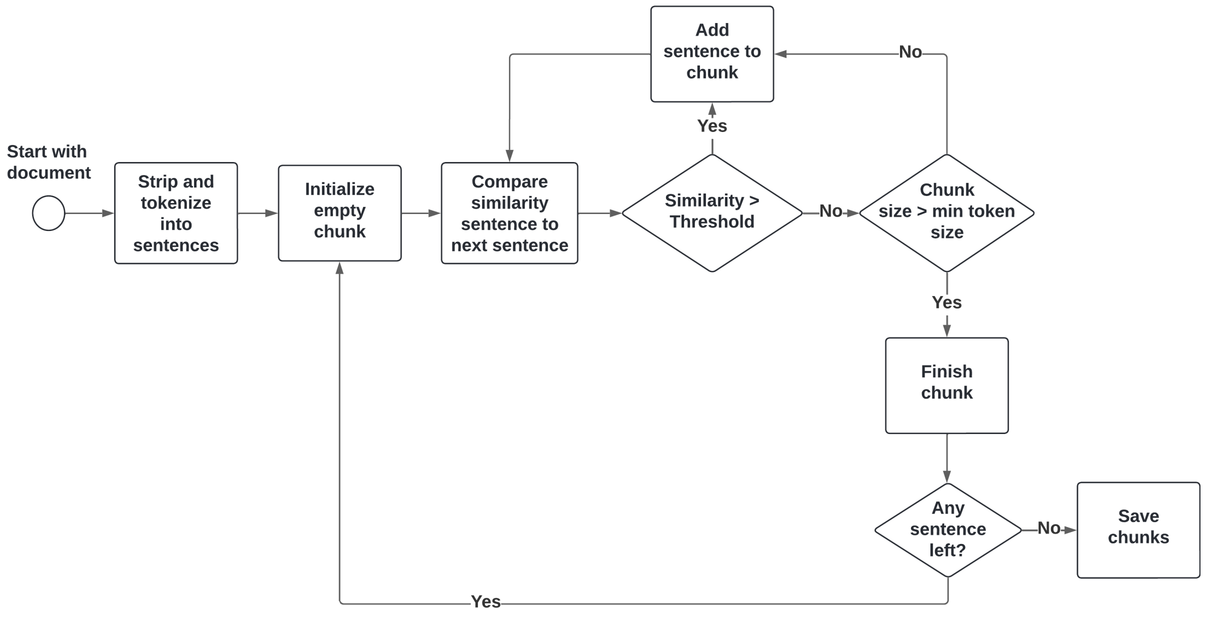


Figure 4: Flow chart of semantic chunking.

4.1.2 Prompt Engineering Techniques

Prompt engineering is done in three ways. The first is done before searching the database. The *MultiQueryRetriever* module from LangChain was applied. This is a retriever that uses the original query to generate k different paraphrased versions that capture the sentiment of the original prompt. Each of these prompts is used to retrieve the relevant document. The results are aggregated, which should give a broader, more relevant set of documents.

The other methods are applied after the documents are retrieved. This is done by asking an LLM to reason step-by-step to improve clarity. Next to that, it also reformulates the original prompt, because the user prompts can be vague. Additionally, it is used to incorporate the context from earlier messages into the prompt to enhance clarity. The instructions for the LLM are shown below.

LLM Prompt: Improving the user query

You are a helpful assistant who reformulates user queries for a retrieval system. Your goal is to generate a standalone, specific version of the user's query.
If context is provided, use it to disambiguate or enrich the question.
Please reason step by step and explain your thought process before giving your final answer.
User Query: {query}
User context: {context_block}
Reformulated Query:

4.1.3 RAG Pipeline Integration

The different RAG pipelines were created. A random model selector chooses a configuration for the model. This process is shown in Figure 5. The configuration is used for the generation of the response of the LLM. Each request is handled by a randomly selected model.

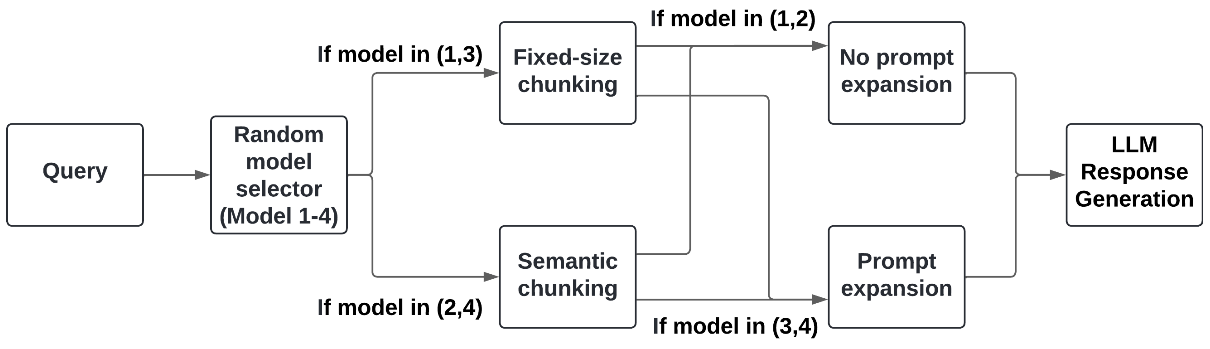


Figure 5: Flow chart of the different RAG pipelines.

4.2 Multiple-Choice Database

A multiple-choice question (MCQ) database was created to evaluate the RAG chatbot with a structured database and a ground truth. The instructions for the LLM to create the database and evaluate the results are discussed.

4.2.1 MCQ Database Generation

An LLM was provided with clusters of related documents from the Leadinfo database, and it was instructed to create MCQs with different difficulties and Bloom levels. This process was repeated with multiple subsets of articles until a database of 100 questions was created. These covered different topics, difficulties, and Bloom levels. The prompt used to generate the database is shown below.

Prompt: Generate MCQs from articles

I want you to generate multiple-choice questions (MCQs) based on a given set of help articles. Each question should be self-contained and understandable without external context, so it can be asked independently (e.g., to another model) without showing the answer options.

For each MCQ, include:
- 1 correct answer
- 3 plausible but incorrect distractors
- The difficulty level (S1–S4, described below)
- The Bloom level (B1–B4, described below)
- The related article(s) used to generate the question (filenames)

Use the following difficulty levels:
- S1: The answer is directly found in a single document.
- S2: The question logically combines multiple facts found in a single document.
- S3: The question combines knowledge from multiple documents in the corpus.

- S4: The answer is not found in the document corpus (to test out-of-distribution detection).

Use the following Bloom levels:
- B1: Remember; Recall facts and basic concepts.
- B2: Understand; Explain ideas or concepts.
- B3: Apply; Use information in new situations
- B4: Analyze; Draw connections among ideas

Distribute the questions as evenly as possible across the four difficulty levels (S1–S4). Each question should be clearly labeled with its difficulty level, and avoid creating all questions at the same difficulty level.

Distribute the questions as evenly as possible across the four Bloom levels (B1–B4). Clearly indicate the Bloom level for each question. Avoid generating all questions at the same Bloom level.

Return the questions and answers in the following JSON format:

```
[
  {
    "difficulty": "S1",
    "question": "What is ...?",
    "correct_answer": "Correct answer here.",
    "incorrect_answers": [
      "Wrong option 1",
      "Wrong option 2",
      "Wrong option 3"
    ],
    "related_articles": [
      "filename1.txt"
    ]
  },
  ...
]
```

Do this for the following sets of documents:
{Set of 5 related documents}

4.2.2 LLM as a Judge

The LLM was instructed to evaluate question-answer pairs based on the aspects of faithfulness, answer relevance, and context relevance. The model GPT-4.1 was used for this, as this is more advanced and better at understanding longer contexts than GPT-4o, which is used for the RAG pipelines [Ope]. The prompt used for the LLM as a judge is:

Prompt: LLM-Based evaluation of RAG answer

You are a helpful evaluator of RAG chatbot responses.

You will be shown:
- A user query
- A chatbot-generated answer
- The context used to generate the answer

Rate the answer on the following 3 aspects:
1. Faithfulness (accuracy of the answer compared to the context)
2. Answer Relevance (how well it answers the user's question)
3. Context Relevance (how useful the context is for answering the question)

Use the 1–5 scale below for each aspect:
1 = Very Poor
2 = Poor
3 = Fair
4 = Good
5 = Excellent

Return your answer in this exact JSON format:

```
{
  "faithfulness": { "score": X, "justification": "..." },
  "answer_relevance": { "score": X, "justification": "..." },
  "context_relevance": { "score": X, "justification": "..." }
}
```

Query: {question}
Chatbot Answer: {answer}
Retrieved Context: {context}

The LLM was also used to evaluate the MCQ database. This was done with the following prompt:

Prompt: Chat-style evaluation of MCQ

```
Your task is to determine which option expresses the same meaning as the given answer.
Answer:
{rag_answer}

Select the most semantically equivalent option from the list below.
Options:
{options}

Reply with only the letter A, B, C, or D.
```

5 Results

The results of the experiment were obtained by asking six Leadinfo employees to evaluate the RAG chatbot. The experiment consisted of the two parts mentioned before: RAG chatbot open questions evaluation and multiple-choice question evaluation. This is analyzed to get an insight into the performance of different RAG configurations and to compare how human and automated LLM evaluations compare.

5.1 Open-Ended Questions Data Exploration

The open questions were asked by six Leadinfo employees, who are domain experts. They were asked to question the RAG chatbot and review each question. Each question-answer pair was simultaneously evaluated by another LLM. This results in a data set of 64 question-answer pairs, evaluated by both a human evaluator and an LLM as a judge. Each question is answered by a random model. Each question-answer pair is reviewed on the aspects of faithfulness, answer relevance, and context relevance. An overview of this is visible in Table 3.

Descriptive statistics, including the means, medians, and standard deviations for each model, provide a clear overview of model performance and human and LLM evaluation, which is visible in table 4a and 4b. These trends are visually represented in figures 6 and 7.

The experiment was done at Leadinfo. Employees were asked to question the RAG chatbot with questions they came across in their daily work. This means that no standard set of open questions was used to evaluate the chatbot. Therefore, the distribution of human evaluators across models can impact the result. Therefore, the distribution of evaluators across models is reviewed. A chi-square test of independence was conducted. This test examined whether the frequency of responses from each user was evenly distributed over the four models. The results indicated no significant association between user and model ($\chi^2(15, N = 64) = 7.75, p = .933$). This suggests that the allocation of users to models was random, and thus, potential user bias is unlikely to have affected the results.

Normality of faithfulness, answer relevance, and context relevance ratings for both LLM and human evaluations were assessed using the Kolmogorov-Smirnov and Shapiro-Wilk tests. In all cases, results indicated significant deviations from normality (all $p < .001$). Despite this, analyses were conducted using ANOVA and MANOVA, as these tests are generally robust to violations of normality with sample sizes above 30, and given that the data are ordinal or categorical, which is common in evaluation studies.

Statistic	**Value**
Total number of questions	64
Unique reviewers (human judges)	6
Responses answered by model 1	14
Responses answered by model 2	18
Responses answered by model 3	17
Responses answered by model 4	17

Table 3: Overview of the Dataset: Question-Answer per model and reviewer distribution

Model	**Faithfulness**			**Answer Relevance**			**Context Relevance**		
	Mean	**SD**	**Median**	**Mean**	**SD**	**Median**	**Mean**	**SD**	**Median**
1	3.71	1.54	4.5	3.93	1.44	4.5	3.64	1.22	4.0
2	4.06	1.26	4.5	4.06	1.26	4.5	3.67	1.37	4.0
3	3.25	1.53	4.0	3.44	1.15	4.0	3.13	1.36	3.0
4	3.19	1.53	3.0	2.94	1.53	4.0	2.75	1.36	3.0

(a) Means, medians, and standard deviations per evaluation metric and per model for human evaluation

Model	**Faithfulness**			**Answer Relevance**			**Context Relevance**		
	Mean	**SD**	**Median**	**Mean**	**SD**	**Median**	**Mean**	**SD**	**Median**
1	4.71	0.61	5.0	4.93	0.27	5.0	4.43	0.94	5.0
2	4.83	0.38	5.0	4.89	0.32	5.0	4.44	1.10	5.0
3	3.81	0.98	4.0	4.31	0.70	4.0	3.19	1.38	3.0
4	4.31	0.98	5.0	4.50	0.16	5.0	3.19	1.38	3.0

(b) Means, medians, and standard deviations per evaluation metric and per model for LLM evaluation

Table 4: Means, medians, and standard deviations per evaluation metric, model, and evaluator

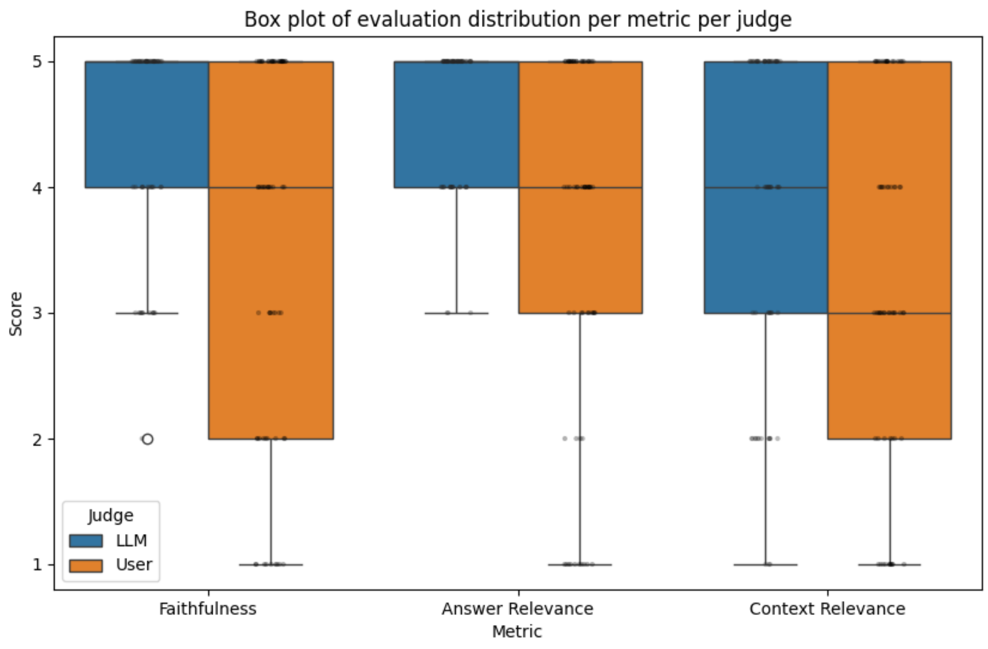


Figure 6: Box plot of evaluation distribution per metric per judge

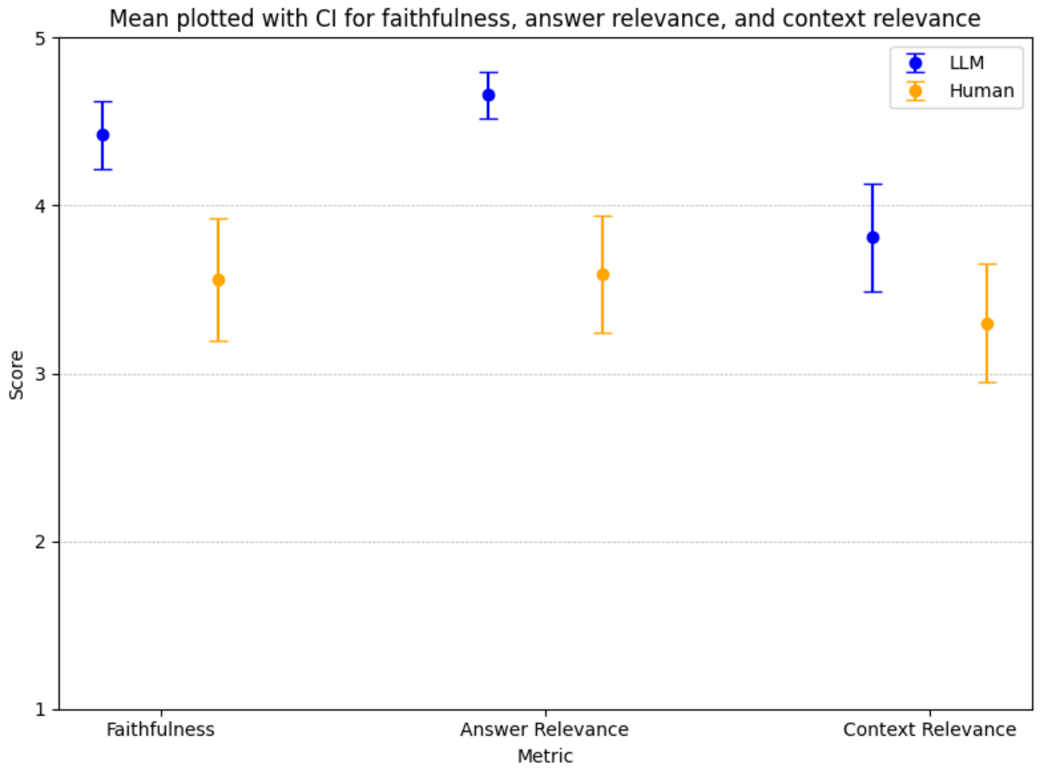


Figure 7: Mean plotted with CI for faithfulness, answer relevance, and context relevance

5.2 Evaluating the RAG Models

A two-way MANOVA with Model (1–4) and Judge (human vs LLM) as fixed factors, visible in Table 5, revealed significant effects for Model (Wilks' $\Lambda = 0.829, p = 0.008$) and Judge (Wilks' $\Lambda = 0.757, p < 0.001$), but not for their interaction (Wilks' $\Lambda = 0.933, p = 0.511$). Univariate follow-ups (Table 6) showed that Model differences persisted on each metric, while Judge effects were especially strong for answer relevance. Post-hoc Tukey tests (Table 7) confirmed that Models 1 and 2 outperformed Models 3 and 4 on all metrics. Figures 8, 9 , and 10 show the estimated marginal means with a 95% CI, where this difference is visible. Thus, the results indicate that models 1 and 2 are generally producing more faithful, answer-relevant, and contextually correct answers than models 3 and 4.

Effect	**Wilks' Λ**	F	**df$_{\text{hyp}}$**	**df$_{\text{err}}$**	p**-value**
Model	0.829	2.56	9	287.33	0.008
Judge	0.757	12.63	3	118	< 0.001
Model $\times$ Judge	0.933	0.92	9	287.33	0.511

Table 5: Results of MANOVA for Model, Judge, and their interaction.

Dependent variable	**Model** F (p)	**Judge** F (p)	**Model** $\times$ **Judge** F (p)
Faithfulness	4.29 (0.007)	17.60 (< 0.001)	0.36 (0.781)
Answer relevance	4.34 (0.006)	33.81 (< 0.001)	0.93 (0.469)
Context relevance	6.17 (< 0.001)	4.95 (0.028)	0.55 (0.648)

Table 6: Tests of between-subjects effects (Type III sums of squares).

Metric	**Comparison**	**Mean diff.** Δ	p**-value**
Faithfulness	Model 2 > Model 3	0.91	0.009
Answer relevance	Model 2 > Model 4	0.75	0.019
Context relevance	Model 1 > Model 4	1.07	0.011
	Model 2 > Model 3	0.90	0.027
	Model 2 > Model 4	1.09	0.005

Table 7: Significant pairwise model differences (Tukey HSD, $\alpha = 0.05$).

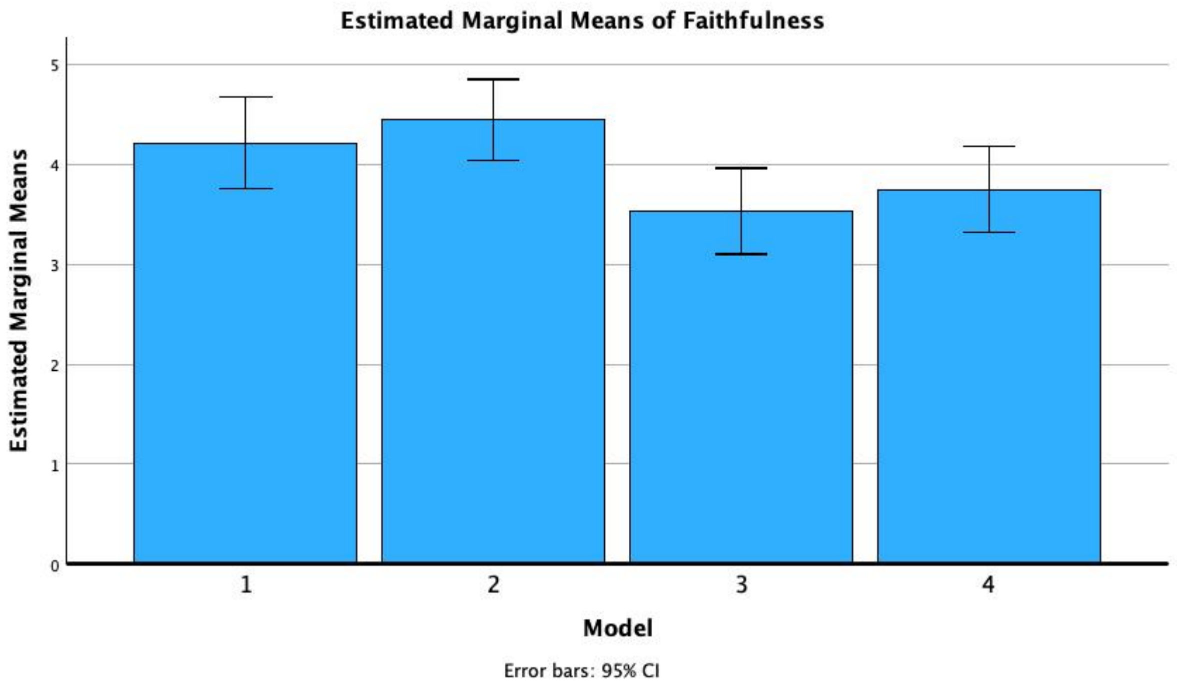


Figure 8: Estimated marginal mean for faithfulness per model

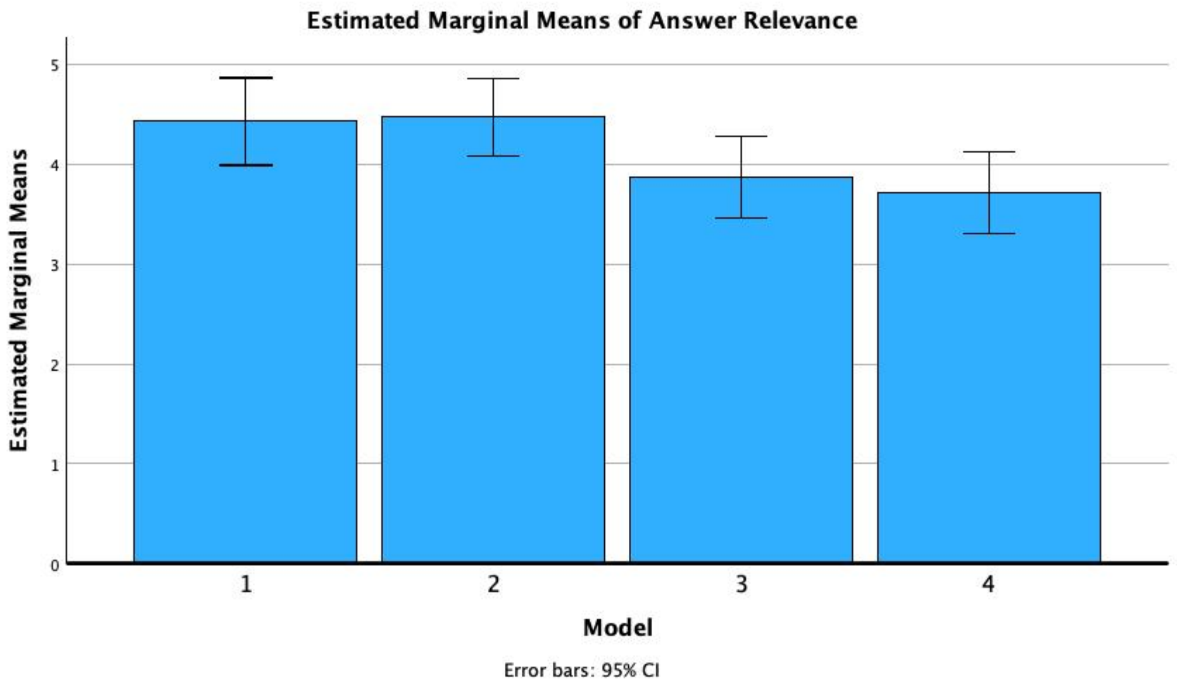


Figure 9: Estimated marginal mean for answer relevance per model

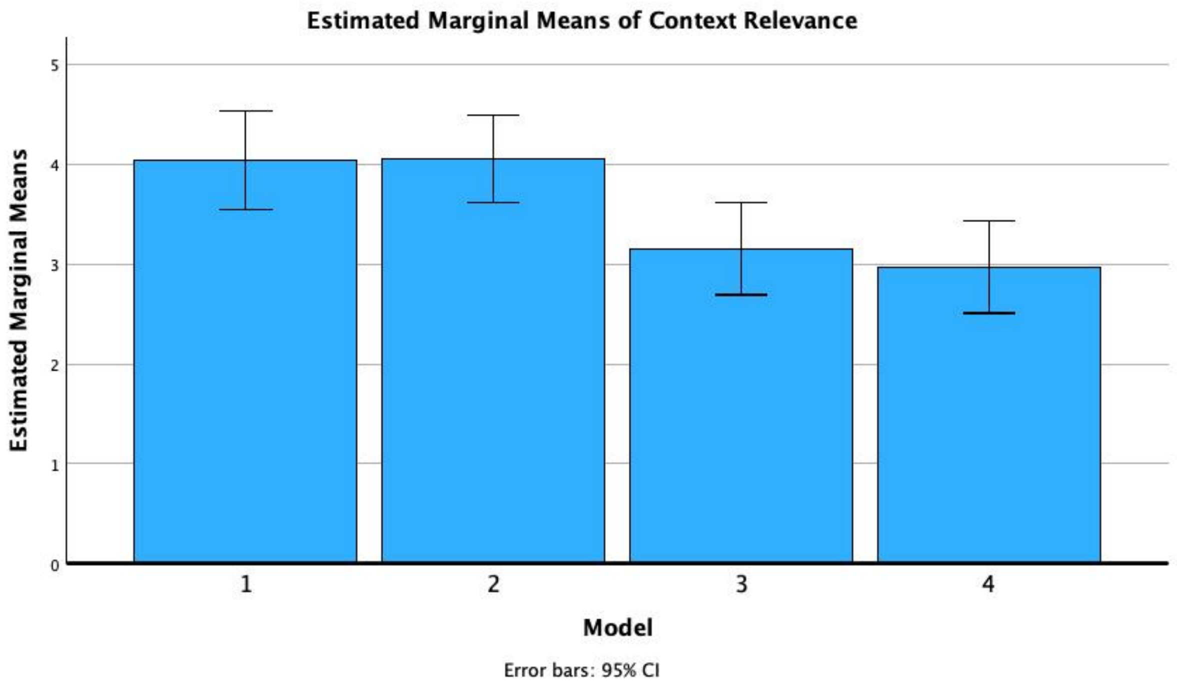


Figure 10: Estimated marginal mean for context relevance per model

5.3 Chunking and Prompt Engineering Effects

To evaluate how both design choices of the RAG chatbot and the type of judge influence the three evaluation metrics, a three-way MANOVA was carried out with Judge (human vs LLM), Chunking strategy (fixed-size vs semantic), and Prompt engineering (none vs expansion) as fixed factors. As shown in Table 8, there was a significant multivariate main effect of Judge (Wilks' $\lambda = .757$, $p < .001$) and Prompt engineering (Wilks' $\lambda = .861$, $p < .001$). Chunking showed no multivariate effect (Wilks' $\lambda = .970$, $p = .304$), and none of the two-way or three-way interactions reached significance (all $p \geq .117$). These results indicate that applying prompt engineering lowers scores across all metrics, and switching from fixed-size to semantic chunks does not influence evaluation outcomes. It also indicates that human raters assign systematically lower scores than the LLM.

Effect	**Wilks'** Λ	F	$\mathbf{df_{hyp}}$	$\mathbf{df_{err}}$	p**-value**
Judge	0.757	12.63	3	118	$< .001$
Chunking	0.970	1.23	3	118	.304
Prompt Engineering	0.861	6.35	3	118	$< .001$
Judge $\times$ Chunking	0.996	0.17	3	118	.918
Judge $\times$ Prompt Engineering	0.951	2.01	3	118	.117
Chunking $\times$ Prompt Engineering	0.995	0.19	3	118	.904
Judge $\times$ Chunking $\times$ Prompt Engineering	0.986	0.54	3	118	.654

Table 8: Results of MANOVA for Judge, Chunking, Prompt Engineering, and their interactions.

Dependent variable	Judge F (p)	Chunking F (p)	Prompt Eng. F (p)
Faithfulness	17.60 (< .001)	0.00 (.988)	10.54 (.001)
Answer relevance	33.81 (< .001)	0.15 (.699)	10.20 (.002)
Context relevance	4.95 (.028)	0.55 (.459)	18.04 (< .001)

Table 9: Univariate ANOVAs for Judge, Chunking, and Prompt Engineering on each metric.

5.3.1 Effect of Chunking Strategies

There was no significant multivariate effect of chunking strategy on faithfulness, answer relevance, or context relevance, which is visible in table 8 (Wilks' $\lambda = .970$, $p = .304$). This indicates that switching from fixed-size chunking to semantic chunking did not lead to a statistically significant improvement or reduction in evaluation scores. This lack of significant difference is also visually reflected in figures 11, 12 and 13, where the estimated marginal means for each evaluation aspect remain similar across the two chunking strategies.

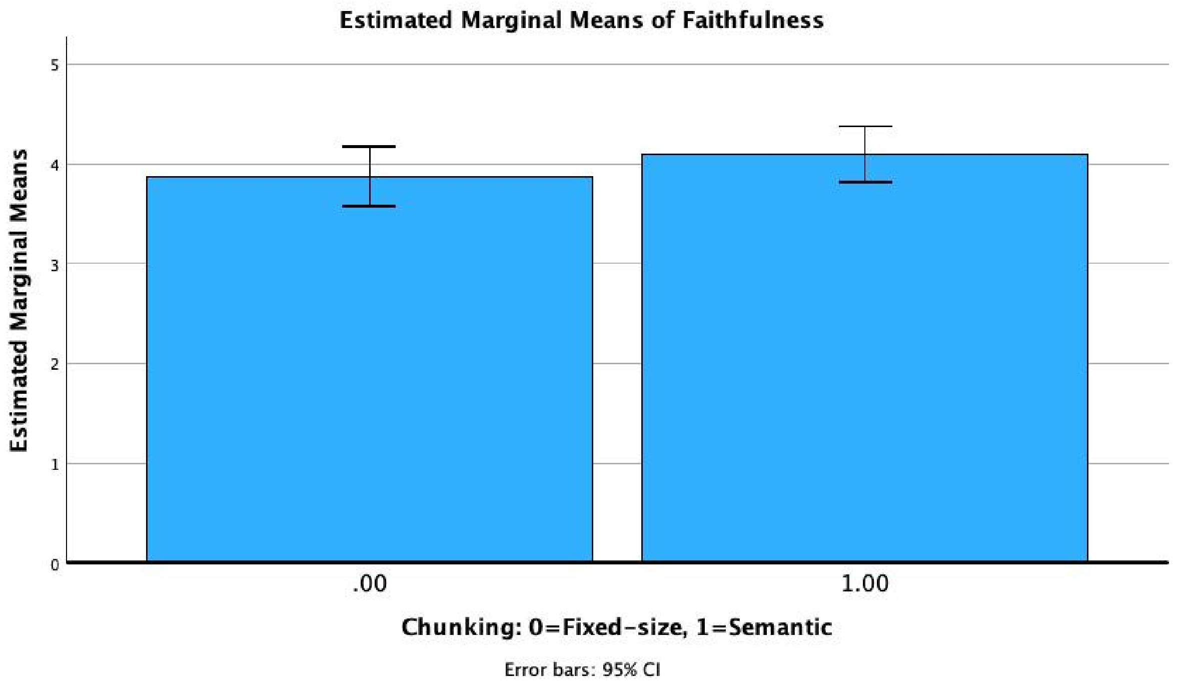


Figure 11: Faithfulness Evaluation (Fixed-size chunking compared to semantic chunking)

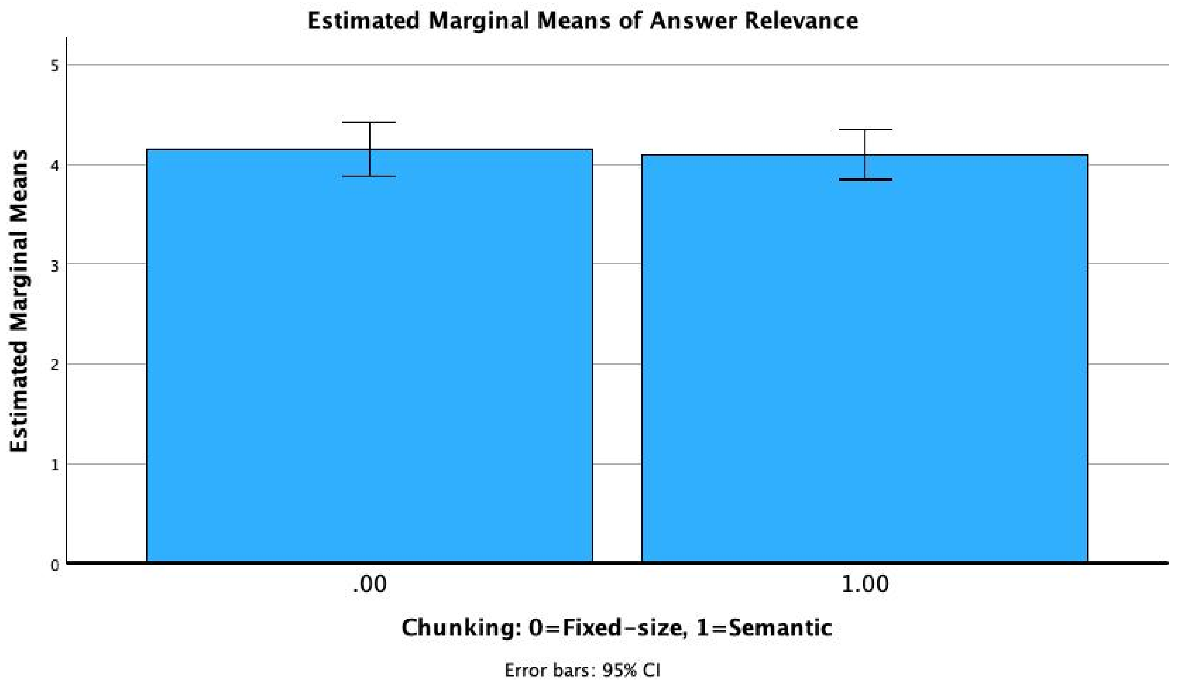


Figure 12: Answer Relevance Evaluation (Fixed-size chunking compared to semantic chunking)

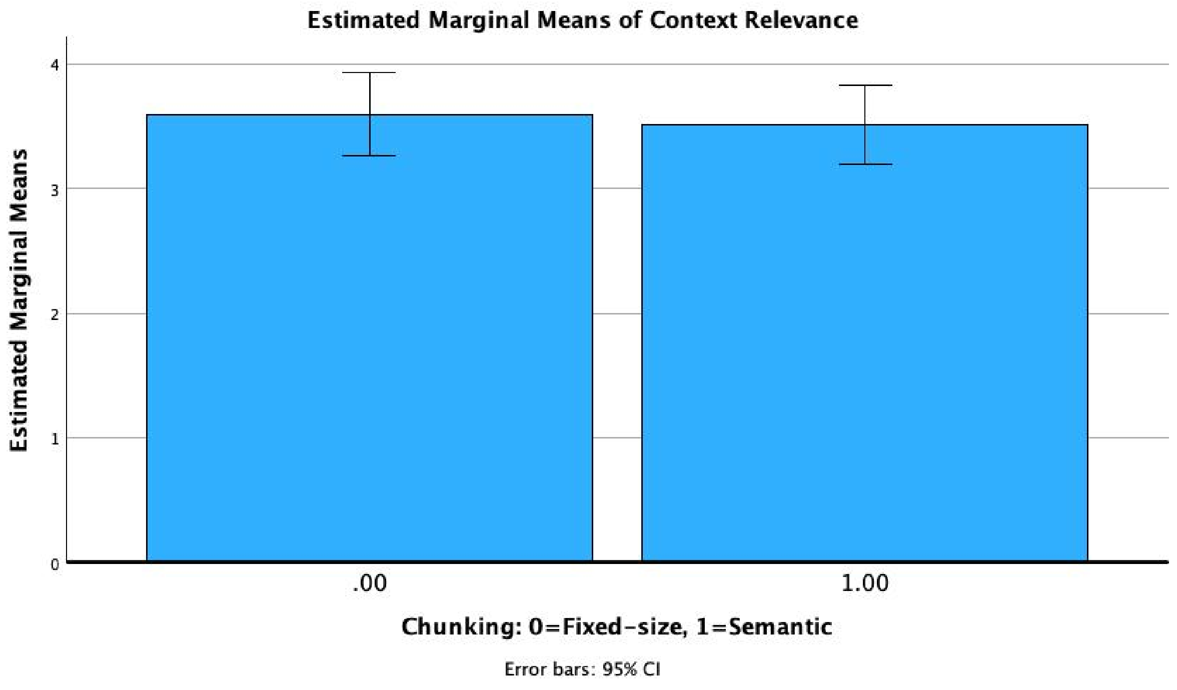


Figure 13: Context Relevance Evaluation (Fixed-size chunking compared to semantic chunking)

5.3.2 Effect of Prompt Engineering Strategies

Prompt engineering has a significant multivariate effect on the evaluation outcomes, which is visible in Table 8 (Wilks' $\lambda = .861$, $p < .001$). The use of prompt engineering led to a statistically significant decrease in scores for faithfulness, answer relevance, and context relevance. Follow-up univariate ANOVAs (Table 9) show that prompt engineering has a significant effect on each metric (all $p < .05$). This significant decrease in scores is also evident in Figures 14, 15, and 16, which show consistently lower estimated marginal means for faithfulness, answer relevance, and context relevance when prompt engineering is applied. Thus, applying prompt engineering has a significantly negative effect on the performance.

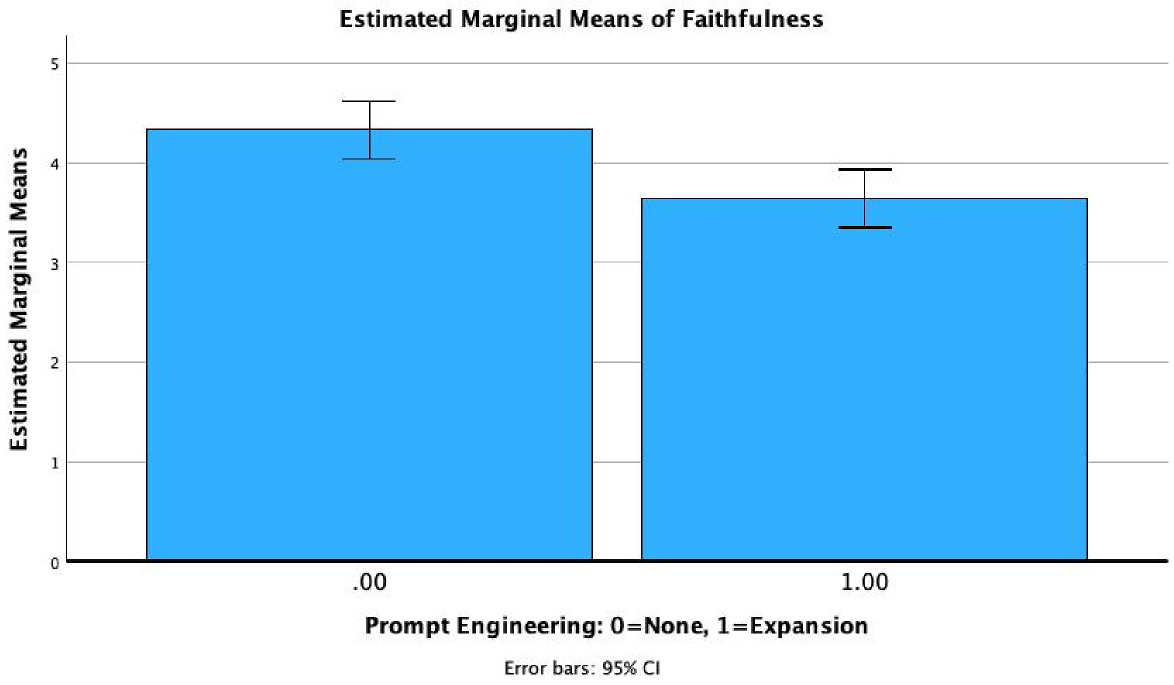


Figure 14: Faithfulness Evaluation (No prompt engineering compared to prompt engineering)

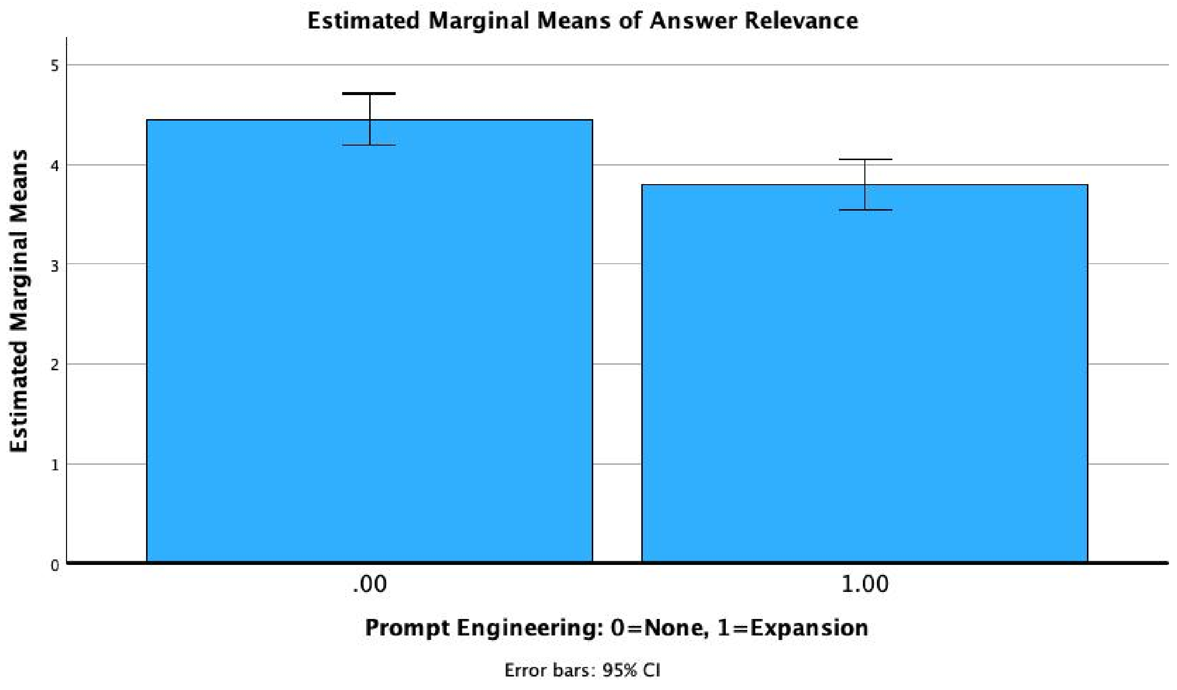


Figure 15: Answer Relevance Evaluation (No prompt engineering compared to prompt engineering)

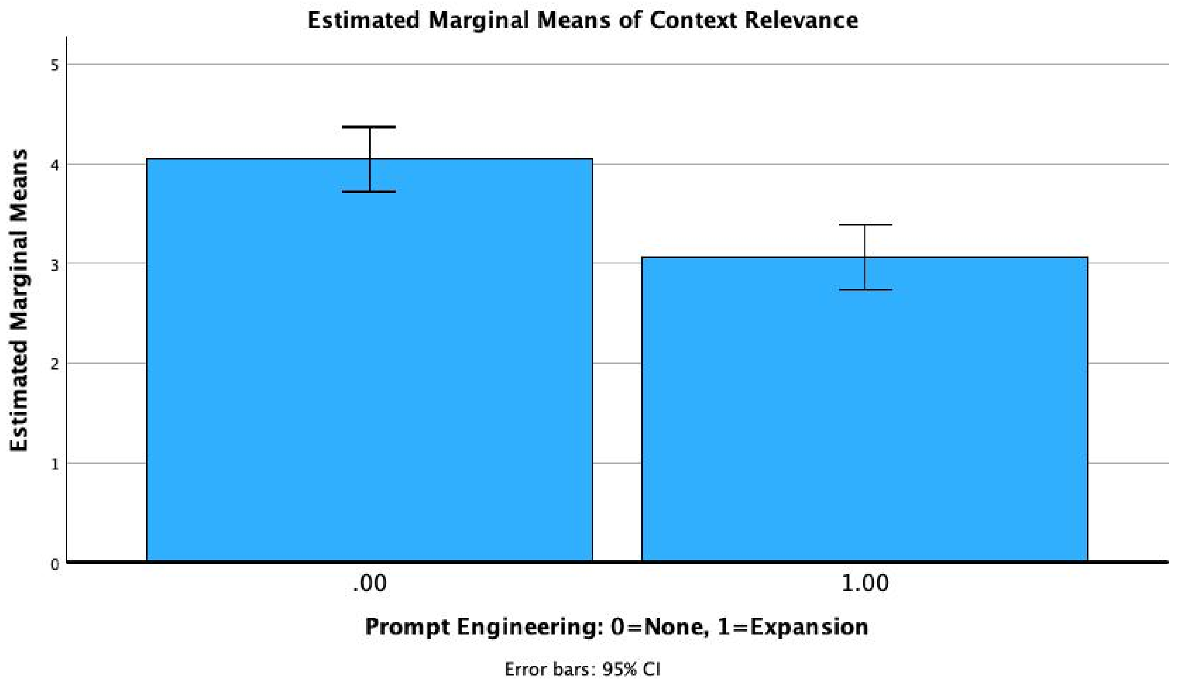


Figure 16: Context Relevance Evaluation (No prompt engineering compared to prompt engineering)

5.4 Human and Automated Evaluation Comparison

A key element of this study is also to compare human and LLM evaluations. This will be done for the open-ended questions and the multiple-choice database.

5.4.1 Open-ended Question Evaluation

Each open question is evaluated by a human and an LLM. To compare the average rating per metric, a paired t-test is done to compare mean scores from the LLM and human users across all three evaluation metrics, which is visible in Table 10. The LLM consistently rated higher than human evaluators for each metric, with all differences being statistically significant ($p < 0.05$). This shows that LLM evaluates responses with higher scores than human raters.

Additionally, the weighted Cohen's kappa coefficients are calculated. The weighted Cohen's kappa is used because that accounts for the ordinal nature of the evaluation metrics, as they are rated on a Likert scale. The results, visible in Table 11, indicate poor, slight, and fair agreement between the human and LLM raters [LK77]. This further reinforces the finding that, despite statistically significant differences in mean scores, LLM evaluations are not aligned with human judgments on an individual item level.

This is visualized in the Figures 17, 18, 19, where scatter plots display human and LLM evaluations across the three criteria. The size of each dot represents the frequency of that specific rating combination. Here it is also visible that there is a low correlation between human and LLM evaluation for answer relevance ($R^2 = 0.155$) and context relevance ($R^2 = 0.140$), and virtually no correlation for faithfulness ($R^2 = 5.85 \times 10^{-6}$)

Metric	LLM Mean	User Mean	n	t-stat	p-value
Faithfulness	4.42	3.56	64	4.05	< 0.001
Answer Relevance	4.66	3.59	64	6.50	< 0.001
Context Relevance	3.81	3.30	64	2.67	0.010

Table 10: Paired t-test results for LLM vs User

Metric	Weighted Cohen's Kappa
Faithfulness	-0.002
Answer Relevance	0.182
Context Relevance	0.348

Table 11: Weighted Cohen's Kappa for agreement between LLM and human raters

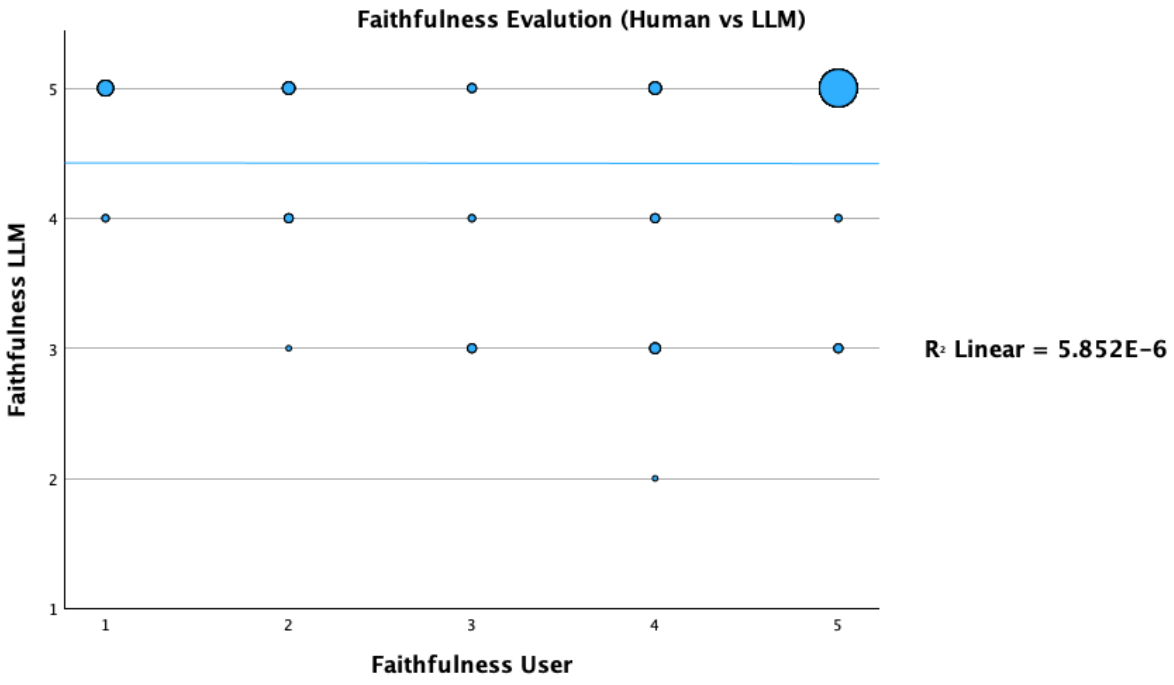


Figure 17: Relationship between human and LLM judgments of faithfulness.

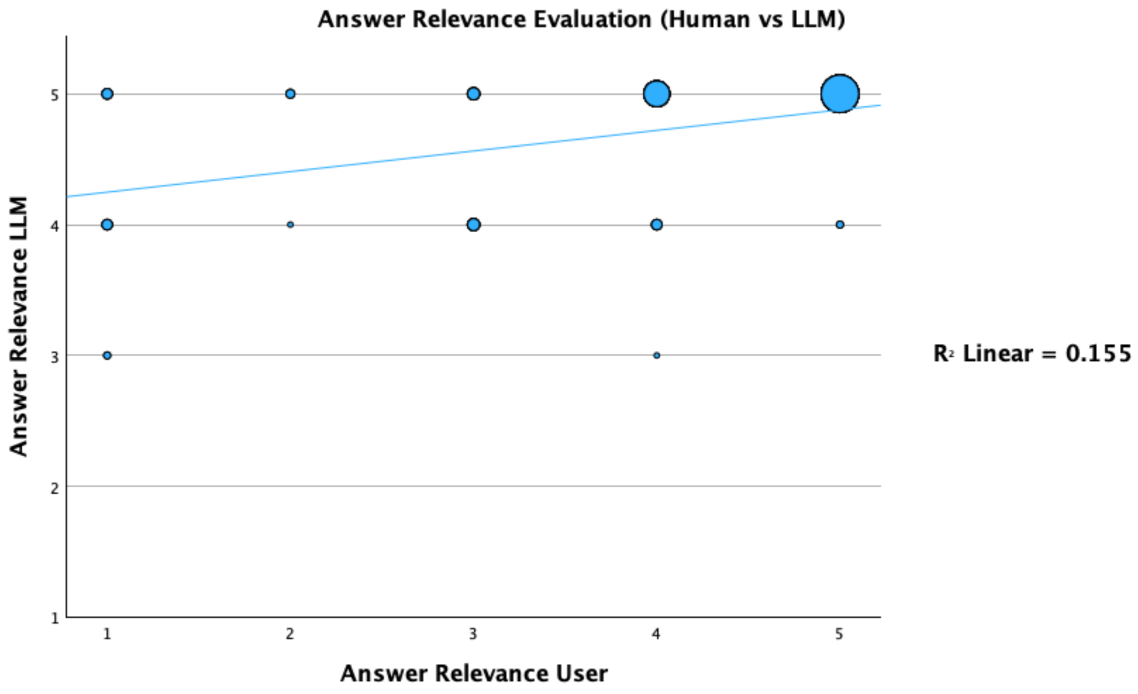


Figure 18: Relationship between human and LLM judgments of answer relevance.

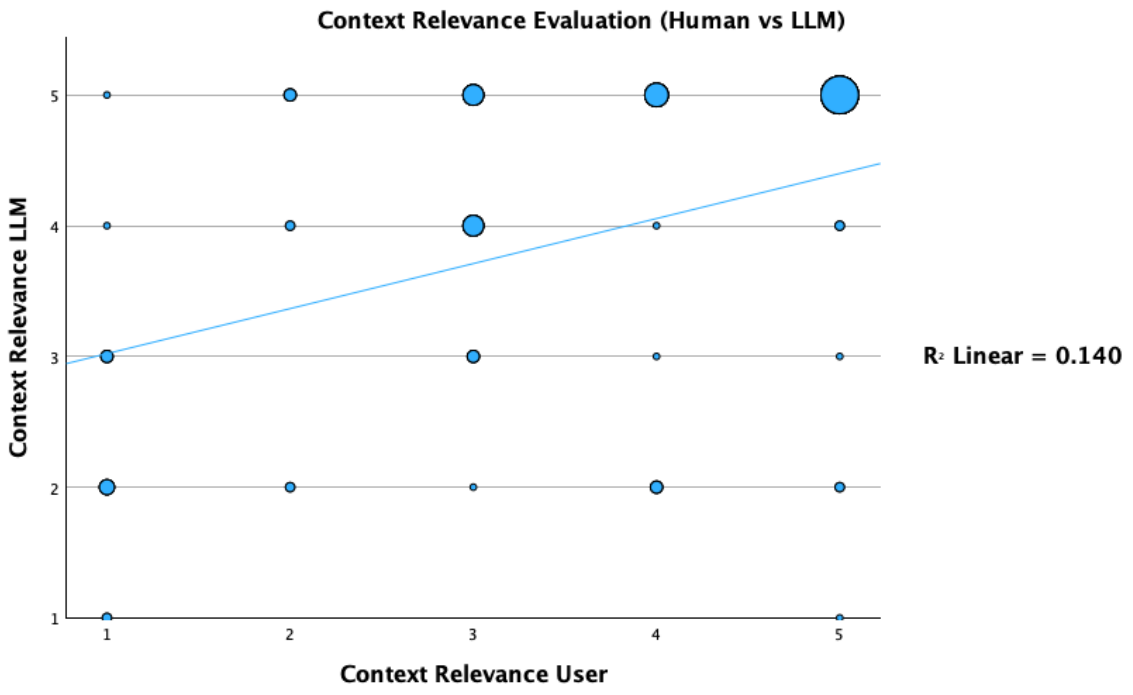


Figure 19: Relationship between human and LLM judgments of context relevance.

5.4.2 Relative Consistency in Evaluation

Section 5.4.1 shows that human and LLM evaluations are statistically different. However, it is also important to know whether the human and LLM evaluations are consistent in the relative ranking of the models. To test this, Spearman's rank correlation coefficients were calculated using the mean scores per model. Table 12 shows that there is a strong correlation for faithfulness and context relevance, and a moderate correlation for answer relevance. However, these results are not statistically significant, due to the limited number of models (N=4).

Metric	Spearman's ρ	p-value
Faithfulness	0.800	0.200
Answer Relevance	0.600	0.400
Context Relevance	0.949	0.051

Table 12: Model-level Spearman's rank correlations between LLM and human mean ratings per model.

At item level, Spearman's rank correlation coefficients were also calculated between human and LLM ratings for each evaluation metric, to see if humans and LLMs agree on an individual level. As shown in Table 13, the correlation for answer relevance was moderate and statistically significant ($\rho = 0.393$, $p = 0.001$), indicating some consistency in the relative ranking of responses between LLM and human raters for this metric. However, for faithfulness ($\rho = 0.103$, $p = 0.418$) and context relevance ($\rho = 0.064$, $p = 0.613$), correlations were weak and not significant, suggesting little to no agreement in the relative ranking of individual responses.

These results contrast with the finding when comparing the means of the different metrics, where a (moderate) strong correlation was found. This shows that humans and LLMs do not agree on an individual level, but align in the overall trend.

Metric	Spearman's ρ	p-value
Faithfulness	0.103	0.418
Answer Relevance	0.393	0.001
Context Relevance	0.064	0.613

Table 13: Item-level Spearman's rank correlation between LLM and human ratings for each metric

5.5 Multiple-Choice Question Database

The multiple-choice database is created to evaluate how human and LLM evaluations align with ground truth. It is also used to evaluate how well a generated multiple-choice question database can be used to evaluate the performance of a RAG system.

5.5.1 Data Exploration

The dataset consists of responses to 100 unique multiple-choice questions. Each question is answered by the RAG chatbot without knowing the answer options. Each evaluator chooses the multiple-choice option that is most similar to the answer of the RAG chatbot. The questions are evaluated by two groups of three humans, each answering 50 questions. Each user was also asked to rate the relevance of the question on a Likert scale, to see how well the database was created. Additionally, the questions are automatically evaluated by an LLM and a cosine similarity baseline. The LLM selects based on semantic equivalence, while cosine similarity compares BERT embeddings of the generated answer with each option.

Table 14 provides an overview of the dataset, including the number of unique questions, total responses, and the distribution across difficulty and Bloom levels.

Characteristic	Count
Number of unique questions	100
Number of human evaluators	6 (2 pairs of 3 - each 50 questions)
Number of automated evaluators	2 (LLM, Cosine - each 100 questions)
Questions per difficulty level	31 / 35 / 21 / 13
Questions per Bloom level	17 / 43 / 30 / 10

Table 14: Overview of the multiple-choice dataset characteristics.

5.5.2 Accuracy per Rater

To see how the quality of rating differed per evaluator, the accuracy of each rater and model was evaluated by calculating the proportion of correct answers out of all questions rated, which is visible in Table 15. Among the human raters, accuracies ranged from 56.0% to 88.0%. The average accuracy across all six human raters was 72.0%. The LLM achieved an accuracy of 81.0%, while the cosine similarity baseline reached 74.0%. These results indicate that the LLM performed comparably to the top-performing human raters and outperformed both the human average and the cosine baseline.

Rater	**Accuracy (%)**
Human average	72.0
LLM	81.0
Cosine	74.0

Table 15: Accuracy per rater and model.

5.5.3 Inter-Rater Agreement

The agreement between raters was evaluated at the question level using Fleiss' kappa, which is appropriate for assessing reliability across multiple rater combinations. The results for all different combinations of evaluators are summarized in Table 16. When considering only human raters, very high agreement was observed ($\kappa = 0.901$ for humans 1–3; $\kappa = 0.822$ for humans 4–6). Adding the LLM or cosine baseline to each human group led to a decrease in kappa values ($\kappa = 0.624$ for humans 1–3 + LLM; $\kappa = 0.594$ for humans 1–3 + Cosine), indicating lower agreement between humans and automated methods. The lowest agreement was seen when both automated methods were included with human raters ($\kappa = 0.549$ for humans 1–3 + LLM + Cosine; $\kappa = 0.451$ for humans 4–6 + LLM + Cosine). Agreement between only the LLM and cosine was lower than among human raters, but still substantial ($\kappa = 0.709$). All kappa values were statistically significant ($p < .001$), indicating agreement exceeded chance.

Raters	**Kappa**	**p-value**
Humans 1–3	0.901	$< .001$
Humans 1–3 + LLM	0.624	$< .001$
Humans 1–3 + Cosine	0.594	$< .001$
Humans 1–3 + LLM + Cosine	0.549	$< .001$
Humans 4–6	0.822	$< .001$
Humans 4–6 + LLM	0.523	$< .001$
Humans 4–6 + Cosine	0.487	$< .001$
Humans 4–6 + LLM + Cosine	0.451	$< .001$
LLM + Cosine	0.709	$< .001$

Table 16: Fleiss' Multirater Kappa (κ) and significance for all combinations of human raters, LLM, and cosine baseline.

5.5.4 Agreement With the Human Majority

The agreement between LLM ratings and the human majority answer was computed for each question, as shown in Table 17. This was done to determine whether the automated evaluators rate like humans. The agreement between the automated evaluation and the human majority answer was calculated using Cohen's kappa, to get a chance-corrected measure of alignment. The LLM achieved a kappa of 0.778 with the human majority, while the cosine similarity baseline achieved a kappa of 0.651. These results indicate substantial agreement and were statistically significant ($p < .001$), while the LLM obtained a higher agreement score. These findings show that both automated evaluators closely reflect the human majority vote, with the LLM outperforming the cosine baseline.

Comparison	**Kappa**	**p-value**
Human majority vs. LLM	0.778	$< .001$
Human majority vs. Cosine	0.651	$< .001$

Table 17: Cohen's kappa (κ) for agreement between the human majority answer and model predictions.

5.5.5 Correlation between Perceived Relevance and Answer Accuracy

Each question in the multiple-choice database was given a relevance score by each human on a Likert scale. To see if more relevant questions are associated with increased correctness, Spearman's rank correlation coefficients were calculated.

As shown in Table 18, only the LLM-selected answers show a small, but statistically significant, positive correlation with relevance ($\rho = .231$, $p = .021$). This shows that the LLM tends to be more accurate when the questions are more relevant. In contrast, the correlations between relevance and correctness for the human-majority and cosine-based answers are not statistically significant.

Comparison	**Spearman's ρ**	**p-value**
Relevance vs. Human Majority	.182	.071
Relevance vs. LLM	.231	.021
Relevance vs. Cosine	.187	.062

Table 18: Spearman's rank correlations between relevance and correctness variables.

5.5.6 Performance as a Function of Difficulty and Bloom Level

All questions in the multiple-choice database were generated based on a difficulty and Bloom level, as described in Section 3.4.1. There are four different difficulty levels and four Bloom levels incorporated in the database. Figure 20 shows the mean accuracy per difficulty level per evaluator. The human majority and cosine evaluation achieved the highest accuracy at the lowest levels, after which accuracy steadily declines. LLM accuracy remains nearly constant across difficulty levels. Figure 21 displays the mean accuracy achieved by each evaluator across the different Bloom levels. The mean accuracy is highest for levels 2 and 3, and lowest in levels 1 and 4. Again, the LLM's evaluations remain relatively stable throughout the different levels.

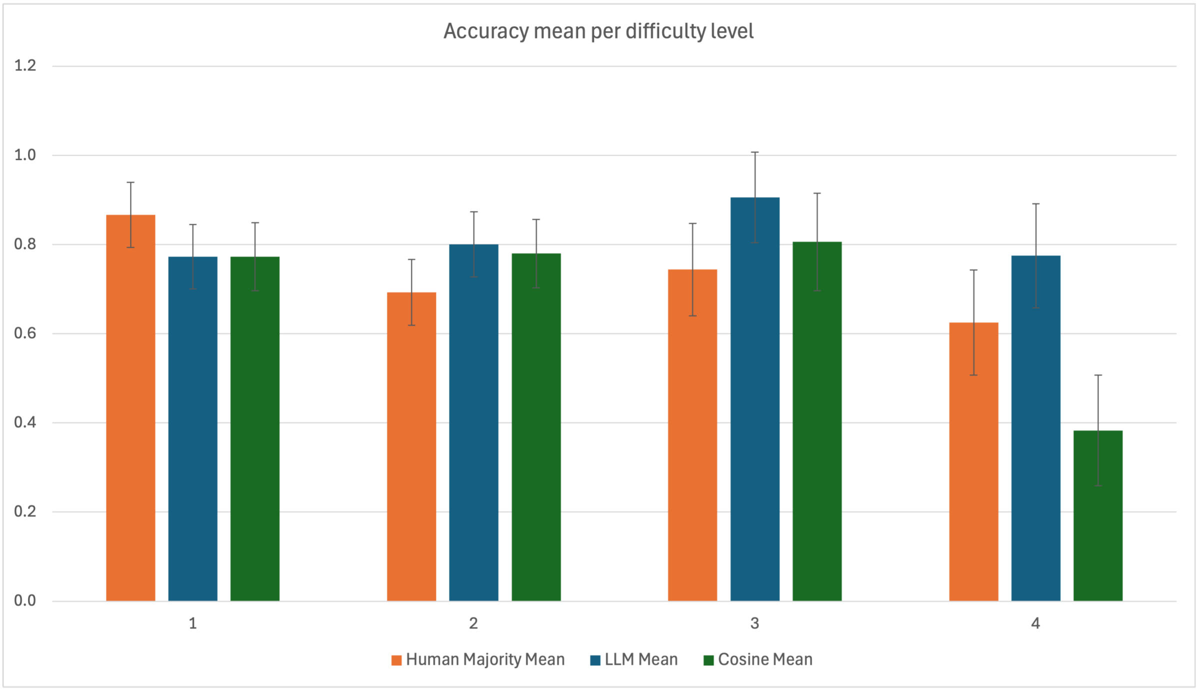


Figure 20: Mean accuracy by difficulty for Human Majority, LLM, and Cosine Baseline. Error bars indicate standard deviation.

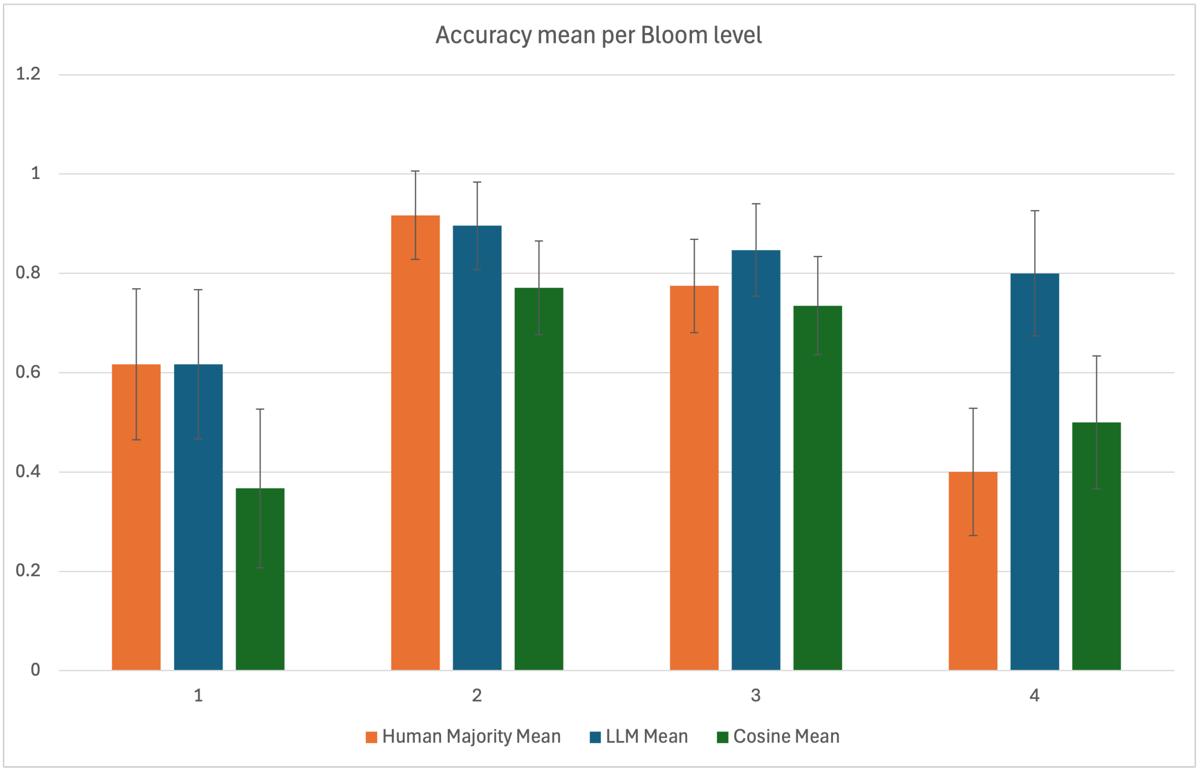


Figure 21: Mean accuracy by Bloom level for Human Majority, LLM, and Cosine Baseline. Error bars indicate standard deviation.

A two-factor MANOVA tested whether the four difficulty levels and the four Bloom levels influenced the three accuracy scores for each evaluator. Although Kolmogorov–Smirnov and Shapiro–Wilk tests showed non-normality for every dependent variable ($p < .001$), MANOVA is robust to such deviations with categorical factors and $n > 30$, so the analysis proceeded. The multivariate test revealed no reliable overall effect of difficulty, Bloom level, or their interaction (all $p > .05$; Table 19).

Separate one-way ANOVAs, however, uncovered two specific patterns (Table 20). For the human-majority scores, accuracy declined significantly with increasing difficulty ($p = .035$) and also differed across Bloom levels ($p = .006$). Tukey's HSD showed that the Bloom effect was driven by a drop from level 2 to level 4 ($p = .009$). None of the six difficulty pairs survived the post-hoc correction, indicating a gradual rather than stepwise decline. For the cosine baseline, difficulty again reached significance ($p = .036$). Post-hoc tests revealed lower accuracy at difficulty level 4 than at levels 1 ($p = .033$) and 3 ($p = .018$). By contrast, neither factor influenced LLM accuracy (all $p > .05$).

Thus, difficulty and Bloom level systematically lead to lower human performance and, to a lesser extent, the cosine baseline. LLM accuracy remains stable. This robustness aligns with the descriptive trends in Figures 20 and 21 and suggests that the LLM performs better with higher difficulty or Bloom levels.

Effect	**Wilks' Λ**	**F**	***p*-value**
Difficulty	0.832	1.87	.058
Bloom	0.829	1.91	.052
Difficulty $\times$ Bloom	0.863	1.48	.156

Table 19: Multivariate tests for effects of difficulty and Bloom levels.

Dependent variable	**Effect**	***F* (df)**	***p*-value**
Cosine Correct	Difficulty	2.96 (3, 90)	.036
Human Majority Correct	Difficulty	2.99 (3, 90)	.035
Human Majority Correct	Bloom	4.37 (3, 90)	.006

Table 20: Summary of significant univariate ANOVA results.

Dependent variable	**Effect**	**Comparison**	**Mean Δ**	**95% CI**	***p*-value**
Cosine	Difficulty	1–4	0.39	0.02–0.76	.033
Cosine	Difficulty	3–4	0.42	0.05–0.77	.018
Human Majority	Bloom	2–4	0.46	0.09–0.83	.009

Table 21: Summary of significant Tukey-HSD pairwise comparisons.

6 Discussion

This research investigated the impact of different chunking strategies and prompt engineering techniques on RAG performance. It also evaluated the alignment between human and automated

LLM evaluation. By looking at both open-ended and multiple-choice scenarios, the findings offer several insights into the implications of deploying RAG pipelines in a business context and how they can be evaluated.

6.1 Key findings

The results make clear that the type of chunking applied did not have a significant impact on chatbot performance (Wilks' $\lambda = .970$, $p = .304$). This suggests that simple chunking strategies may be sufficient for business applications. There may be limited benefit to investing in more complex alternatives for comparable situations. On the other hand, prompt engineering had a noticeable negative effect on the key metrics of faithfulness, answer relevance, and context relevance (Wilks' $\lambda = .861$, $p < .001$). Instead of improving performance, more elaborate prompts seemed to introduce ambiguity or distract from the main query, leading to lower scores. Applying prompt engineering must be thoroughly evaluated, as it gave worse results in this experiment. However, as prompt engineering tactics were bundled rather than tested in isolation, no conclusions can be drawn on which specific technique caused the decline, or whether some might actually be beneficial on their own.

Comparing human and automated LLM evaluation revealed both similarities and key differences. LLMs consistently gave higher ratings to chatbot responses than human evaluators did for all metrics (all paired t-tests $p < .01$). Agreement at the response level was low, with weighted Cohen's κ ranging from $-.00$ to .35, which means no to only fair agreement between human and LLM scores. This means that human and LLM evaluations cannot be directly compared, as they gave different results in this experiment.

Despite this, the relative ordering of models on average showed a moderate to strong correlation (Spearman's ρ between .600 and .949) between human and LLM ratings, suggesting that while the two approaches disagree on specifics, they do agree on overall trends of models' performance. However, this result was not significant (all $p > .05$), so no conclusions can be drawn from this.

The analysis of the multiple-choice dataset shows the potential of automated evaluation. The LLM (81%) outperformed the average human (72%) in accuracy, and even aligned closely with the human majority in many cases. The cosine similarity baseline, while useful for quick comparisons, did not achieve the same level of alignment. These results point to the value of LLMs for scalable and efficient evaluation in tasks with a ground truth. Although caution is still needed when the questions are more difficult or advanced.

6.2 Implications

These findings suggest that investing heavily in advanced chunking might not give a significantly different result, and that prompt engineering should be approached carefully. Simple, well-structured prompts and chunking strategies may often suffice.

Automated evaluation by an LLM shows its potential when used carefully. For evaluating open questions, LLMs tend to rate significantly higher than humans, which is a risk. Using solely automated LLM evaluation tasks without a ground truth might lead to wrong findings about the system's performance. It could potentially be used to compare performance, as the relative ranking of models is moderately to strongly correlated with human ranking.

LLMs show clear potential for evaluating tasks with a clear ground truth, such as multiple-choice questions. They show to be more accurate than human judgment or cosine comparison, and deliver results quickly. However, LLMs can lack critical judgment. Therefore, organizations should combine automated and human evaluations to ensure both efficiency and nuanced assessment.

6.3 Limitations

Several limitations should be considered when interpreting these results. The study was conducted within a single organization and on a relatively limited dataset, which may affect generalizability. Additionally, due to the limited scope and high cost of human evaluation by domain experts, the number of question-answer pairs evaluated was limited, as there were 64 question-answer pairs.

We did not compare the LLM-judge scores with automated suites such as RAGAS or ARES. Benchmarking against those metrics is left for future work.

The open-ended nature of the questions and the subjectivity in human ratings introduce potential biases, and there is a risk that findings are at least partly shaped by the particular context and data at hand. Additionally, due to the scope of the project, the employees were free to determine which questions they wanted to ask the RAG system. This was done to get a more diverse set of questions evaluated, which is more representative in a business setting. However, this means there is no standard set of open questions used, nor did we check that proper content was available in principle to answer the questions. Therefore, the quality of questions and ratings could have differed between users. This could have potentially impacted the evaluations.

The multiple-choice database was completely generated by an LLM. This has several advantages, but it also means that the ground truth is not generated by human experts. This could potentially mean that not all ground truth is true. An LLM could also encode bias in the ground truth, which could improve the accuracy of the LLM evaluator. In addition, while the multiple-choice database was designed to cover a range of difficulty levels and cognitive demands, the actual distribution was uneven, in part due to the constraints of both the source material and the LLM's output. Using a structured (MCQ) database might also be less representative of a real-world application.

The MCQ dataset was not used to compare individual RAG configurations, because each model contributed only around 25 answers, and the automatically generated ground truths were not independently validated. A meaningful comparison would require either a larger MCQ set or human verification of all ground truths.

6.4 Future Research

This research provides several opportunities for further research on RAG design choices and on different types of (automated) LLM evaluation of RAG systems.

6.4.1 RAG Design Choices

Further research could extend these findings by testing additional chunking strategies and by expanding the scope to include a broader variety of domains and datasets. Prompt engineering could also be evaluated more thoroughly. In this experiment, keyword expansion and query reformulation were applied. The separate effects and different methods could be evaluated to see the impact of individual prompt engineering techniques.

6.4.2 Automated Evaluation of Open Questions

The results show that an LLM evaluates with higher scores than a human. Research can be done on tuning an LLM so that it evaluates more critically and human-like. For example, by calibrating the LLM with, for example, rubric examples or adding more context. Once calibrated, the LLM judge can be benchmarked against automated evaluation frameworks such as RAGAS and ARES.

Although human and LLM evaluations on open questions significantly differ, they are similar in relative ranking. However, due to the small size of the experiment, the results were not significant. Research should be done on a larger scale with more model configurations to see if the relative ranking of human and LLM aligns, as this would be a valuable way to automatically evaluate different RAG configurations.

6.4.3 Automated Evaluation With a Ground Truth

As seen in this research, evaluating with a ground truth gave better results. Therefore, more research could be done on efficiently and effectively creating or generating databases with ground truths, as this currently requires much effort. Future work should automate RAG assessment by building a reusable MCQ or structured QA dataset template. This can be used with a domain-specific database to automatically create a MCQ or structured QA dataset with minimal effort.

High-quality ground truth is essential for quality assessment. Therefore, research can also be done on different methods of creating datasets and the effect on the quality of the ground truths. Different types of datasets, such as MCQ or structured databases, and different design methods, such as manual, hybrid, or fully automated, can be compared. The goal is to create scalable pipelines that maintain human-level precision while reducing cost.

6.4.4 Hybrid Evaluation

Open-ended questions better mirror daily usage but make evaluation difficult and costlier. Structured datasets are more suitable for automated LLM evaluation, but are less representative of real-world scenarios. Therefore, future research should look into hybrid evaluation.

This can be done by tuning an LLM so that it creates most of the questions and answers and flags the low-confidence ones for manual review. Experiments should measure the reliability and time savings of this method.

Another method would be for domain experts to create a small, high-quality reference set of question-answer pairs. This would serve as an example of correct, complete, and well-grounded answers for the LLM. Then, it should be measured if this improves the LLM evaluation significantly.

7 Conclusion

This thesis set out to investigate how chunking, prompt engineering, and evaluation methods impact the performance and assessment of RAG chatbots. The results show that simple chunking is just as effective as more complex methods, while prompt engineering can lower performance if not applied carefully. Automated evaluation using LLMs works well for structured tasks but still differs from human judgment in open-ended cases. Overall, the best results are achieved by pairing simple design choices with a balanced mix of human and automated evaluation. Future work should focus on refining these approaches, especially hybrid evaluation, for even better reliability and practical use.

References

[AAA$^+$23] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[BH17] Katherine A Batterton and Kimberly N Hale. The Likert scale what it is and how to use it. *Phalanx*, 50(2):32–39, 2017.

[BSB25] Lorenz Brehme, Thomas Ströhle, and Ruth Breu. Can LLMs Be Trusted for Evaluating RAG Systems? A Survey of Methods and Datasets. *arXiv preprint arXiv:2504.20119*, 2025.

[CHGD24] Zina Chkirbene, Ridha Hamila, Ala Gouissem, and Unal Devrim. 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)*, pages 229–234. IEEE, 2024.

[DGD$^+$24] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. *arXiv preprint arXiv:2401.08281*, 2024.

[EJAS24] Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. Ragas: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158, 2024.

[FDN$^+$24] Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6491–6501, 2024.

[For10] Mary Forehand. Bloom's taxonomy. *Emerging perspectives on learning, teaching, and technology*, 41(4):47–56, 2010.

[GXG$^+$23] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2, 2023.

[L$^+$24] Benoît Lu et al. Evaluating LLMs on large contexts: a RAG approach on text comprehension. 2024.

[Lea25a] Leadinfo. Leadinfo - B2B Website Visitor Recognition. https://www.leadinfo.com/en/, 2025. Accessed: 2025-03-04.

[Lea25b] Leadinfo. Leadinfo Help Centre. https://help.leadinfo.com/en, 2025. Accessed: 2025-05-05.

[LK77] J Richard Landis and Gary G Koch. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174, 1977.

[LPP+20] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[Not25] Notion Labs, Inc. Notion – Your connected workspace for wiki, docs & projects. https://www.notion.com/, 2025. Accessed: 2025-05-05.

[Ope] OpenAI. Compare Models. https://platform.openai.com/docs/models/compare. Accessed: 2025-04-06.

[Ope24] OpenAI. GPT-4o. https://platform.openai.com/docs/models/gpt-4o, 2024. Accessed: 2025-06-01.

[PNEF24] Zhiyuan Peng, Jinming Nian, Alexandre Evfimievski, and Yi Fang. ScopeQA: A Framework for Generating Out-of-Scope Questions for RAG. *arXiv preprint arXiv:2410.14567*, 2024.

[SBK+24] Shangeetha Sivasothy, Scott Barnett, Stefanus Kurniawan, Zafaryab Rasool, and Rajesh Vasa. RAGProbe: An Automated Approach for Evaluating RAG Applications. *arXiv preprint arXiv:2409.19019*, 2024.

[SFKPZ23] Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. Ares: An automated evaluation framework for retrieval-augmented generation systems. *arXiv preprint arXiv:2311.09476*, 2023.

[Sla25a] Slack Technologies, LLC. Slack. https://slack.com/, 2025. Accessed: 2025-06-01.

[Sla25b] Slack Technologies, LLC. Slack AI Chatbot: Bolt for Python Sample. https://github.com/slack-samples/bolt-python-ai-chatbot, 2025. Accessed: 2025-06-01.

[SSS+24] Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*, 2024.

[SZ24] Mingyang Song and Mao Zheng. A Survey of Query Optimization in Large Language Models. *arXiv preprint arXiv:2412.17558*, 2024.

[TA23] Oguzhan Topsakal and Tahir Cetin Akinci. Creating large language model applications utilizing langchain: A primer on developing llm apps fast. In *International Conference on Applied Engineering and Natural Sciences*, volume 1, pages 1050–1056, 2023.

[TY24] Yixuan Tang and Yi Yang. Multihop-rag: Benchmarking retrieval-augmented generation for multi-hop queries. *arXiv preprint arXiv:2401.15391*, 2024.

[VKS10] Susana M Vieira, Uzay Kaymak, and João MC Sousa. Cohen's kappa coefficient as a performance measure for feature selection. In *International conference on fuzzy systems*, pages 1–8. IEEE, 2010.

[WD20] Jiapeng Wang and Yihong Dong. Measurement of text similarity: a survey. *Information*, 11(9):421, 2020.

[WNM$^+$19] Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang. Multi-passage BERT: A globally normalized BERT model for open-domain question answering. *arXiv preprint arXiv:1908.08167*, 2019.

[XCJS24] Yunqi Xu, Tianchi Cai, Jiyan Jiang, and Xierui Song. Face4RAG: Factual Consistency Evaluation for Retrieval Augmented Generation in Chinese. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6083–6094, 2024.

[YGZ$^+$24] Hao Yu, Aoran Gan, Kai Zhang, Shiwei Tong, Qi Liu, and Zhaofeng Liu. Evaluation of retrieval-augmented generation: A survey. In *CCF Conference on Big Data*, pages 102–120. Springer, 2024.

[Yin24] Yingbiao. Chatbot with LLM and RAG in Action. https://medium.com/@yingbiao/chatbot-with-llm-and-rag-in-action-575382df4323, 2024. Accessed: 2025-04-18.

[ZHS$^+$24] Shengming Zhao, Yuheng Huang, Jiayang Song, Zhijie Wang, Chengcheng Wan, and Lei Ma. Towards understanding retrieval accuracy and prompt quality in rag systems. *arXiv preprint arXiv:2411.19463*, 2024.

[ZJF$^+$24] Jihao Zhao, Zhiyuan Ji, Yuchen Feng, Pengnian Qi, Simin Niu, Bo Tang, Feiyu Xiong, and Zhiyu Li. Meta-chunking: Learning efficient text segmentation via logical perception. *arXiv preprint arXiv:2410.12788*, 2024.

[ZLX$^+$24] Kunlun Zhu, Yifan Luo, Dingling Xu, Yukun Yan, Zhenghao Liu, Shi Yu, Ruobing Wang, Shuo Wang, Yishan Li, Nan Zhang, et al. Rageval: Scenario specific rag evaluation dataset generation framework. *arXiv preprint arXiv:2408.01262*, 2024.

[ZW24] Kui Zhao and Guoyu Wang. Keyword-Enhanced Semantic Retrieval and Multi-Dimensional Relevance Ranking in RAG. In *2024 10th International Conference on Computer and Communications (ICCC)*, pages 464–468. IEEE, 2024.